RICE UNIVERSITY

Dimension Reduction Methods with Applications to High Dimensional Data with a Censored Response

by

Tuan S. Nguyen

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

APPROVED, THESIS COMMITTEE:

12/1/09

Javier Rojo, Chair and Advisor
Professor of Statistics

Rudy Guerra
Professor of Statistics

Yin Zhang
Professor of Computational and Applied Mathematics

Houston, Texas

December, 2009

ABSTRACT

Dimension Reduction Methods with Applications to High Dimensional Data with a Censored Response

by

Tuan S. Nguyen

Dimension reduction methods have come to the forefront of many applications where the number of covariates, p, far exceed the sample size, N. For example, in survival analysis studies using microarray gene expression data, 10-30K expressions per patient are collected, but only a few hundred patients are available for the study. The focus of this work is on linear dimension reduction methods. Attention is given to the dimension reduction method of Random Projection (RP), in which the original $p-$dimensional data matrix X is projected onto a $k-$dimensional subspace using a random matrix Γ. The motivation of RP is the Johnson-Lindenstrauss (JL) Lemma, which states that a set of N points in $p-$dimensional Euclidean space can be projected onto a $k \geq \frac{24 \ln N}{3\epsilon^2 - 2\epsilon^3}$ dimensional Euclidean space such that the pairwise distances between the points are preserved within a factor $1 \pm \epsilon$. In this work, the JL Lemma is revisited when the random matrix Γ is defined as standard Gaussian and Achlioptas-typed. An improvement on the lower bound for k is provided by working directly with the distributions of the random distances rather than resorting to the moment generating function technique used in the literature. An improvement on the lower bound for k is also provided when using pairwise L_2 distances in the space of the original points and pairwise L_1 distances in the space of the projected points.

Another popular dimension reduction method is Partial Least Squares. In this work, a variant of Partial Least Squares is proposed, denoted by Rank-based Modified Partial Least Squares (RMPLS). The weight vectors of RMPLS can be seen to be the solution to an optimization problem. The method is insensitive to outlying values of both the response and the covariates, and takes into account the censoring information in the construction of its weight vectors. Results from simulation and real datasets under the Cox and Accelerated Failure Time (AFT) models indicate that RMPLS outperforms other leading methods for various measures when outliers are present in the response, and is comparable to other methods in the absence of outliers in the response.

List of Abbreviations

In parentheses are the sections in which the abbreviations are discussed.

AFT model: Accelerated Failure Time Model (section 4.2.1)

ave(*bias*): Bias of the Estimated Survival Function Evaluated at the Average of the Covariates (section 4.2.5)

ave(*bias.ind*): Bias of the Estimated Survival Function Evaluated Using the Covariates of the Individuals (section 4.2.5)

ave(*ds*): Mean Squared Error of the Estimated Survival Function Evaluated at the Average of the Covariates (section 4.2.5)

ave(*ds.ind*): Mean Squared Error of the Estimated Survival Function Evaluated using the Covariates of the Individuals (section 4.2.5)

cdf: Cumulative Distribution Function (section 3.1)

Cox PH model: Cox Proportional Hazards Model (section 4.2.1)

CPCR: Correlation Principal Component Regression (section 2.5)

CV: Cross Validation (section 4.2.4)

JL Lemma: Johnson Lindenstrauss Lemma (section 3.2)

KM: Kaplan-Meier Estimator of the Survival Function (section 4.2.1)

mgf: Moment Generating Function (section 3.2)

$min(CV(fit.error))$: Minimized Cross Validation of the Squared Residuals or Fit Error (section 4.2.4)

$min(CV(surv.error))$: Minimized Cross Validation of the Squared Error of the Estimated Survival Function (section 4.2.4)

$MSE(\beta)$: Mean Squared Error of the Estimated Weights on the Genes (section 4.2.5)

MPLS: Modified Partial Least Squares (section 2.7.1)

MIPLS: Mean Imputation Partial Least Squares (section 2.7.2)

NA: Nelson-Aalen Estimator of the Survival Function (section 4.2.1)

PARD: Percentage of Additional Reduction in Dimension (section 3.5.2)

PCA: Principal Component Analysis (section 2.2)

PC: Principal Component (section 2.2)

pdf: Probability Distribution Function (section 4.2.1)

PLS: Partial Least Squares (section 2.7)

RMIPLS: Rank-based Mean Imputation Partial Least Squares (section 4.1)

RMPLS: Rank-based Modified Partial Least Squares (section 4.1)

RRWPLS: Rank-based Reweighted Partial Least Squares (section 4.1)

RWPLS: Reweighted Partial Least Squares (section 2.7.2)

SIR: Sliced Inverse Regression (section 2.6)

SPCR: Supervised Principal Component Regression (section 2.4)

TVPE: Total Variation of the Predictor Explained (section 4.2.6)

UNIV: Univariate Selection (section 2.3)

Acknowledgements

The completion of this thesis would not have been possible without the support, guidance, and encouragement of several individuals to whom I would like to extend my sincere appreciation and gratitude. First, I would like to thank my thesis advisor, Dr. Javier Rojo, for his patience and supervision over the past four and a half years, his expertise and extensive reviews of my research and paper writing process, and his integrity and work ethic from which I have learned and grown personally as well as professionally.

I would like to thank two other members of my thesis committee, Dr. Rudy Guerra and Dr. Yin Zhang, for taking their valuable time to read this thesis and making helpful comments and suggestions.

I would like to thank my colleagues in the Statistics Department for their support and encouragement. Special thanks to Beibei Guo, Alejandro Cruz-Marcelo and Dajiang Liu for numerous helpful discussions regarding this thesis.

On a personal note, I would like to thank my parents, my older brother and two older sisters for their love, support and devotion over the years, especially during my graduate career. Without them, I would not have been the person I am today. I would like to extend my gratitude to my two close friends Hien Nguyen and Bill Tran for their encouragement and support.

Support

This thesis work is partially supported by NSF Grant SES-0532346, NSA RUSIS Grant H98230-06-1-0099, and NSF REU Grant MS-0552590.

Contents

Illustrations

Tables

Introduction

Dimension reduction methods play an important role in many applications with the arrival of the "small N, large p" paradigm. One such application is in the field of biomedical research using survival analysis with microarray data, where only a few hundred of patients are available for study but 10-30K gene expression levels per patient are collected. Microarrays allow researchers to quickly and efficiently perform simultaneous analysis of thousands of genes in a single experiment by providing extensive and valuable information on the gene function. Much emphasis of microarray analysis has been placed on discovering or identifying the gene expressions that relate to biological processes or diseases, classifying gene expression data into categories such as types and severity of tumors, and studying the interactions among the genes. However, because microarray data often include patients' survival information, it is of interest to analyze censored patient survival times (response) taking into account their corresponding gene expression levels (covariates). However, the ability to measure large number of genes in a single experiment has also resulted in data with the number of genes, p, far exceeding the number of patients (cases), N. The high-dimensionality of the microarray data needs to be reduced before embarking on any type of statistical analysis.

Dimension reduction seeks to reduce the dimension of the microarray dataset, often in the order of thousands, while trying to retain most of the relevant information contained in the original dataset. However, dimension reduction of microarray data

has not effectively found a low-dimensional projection that provides an accurate representation of the original data (Kharal [58]). The dimension reduction methods are usually data-specific, i.e. one method may be better than another for one dataset, but the reverse maybe true for another dataset.

In the context of survival analysis, one popular regression model that takes into account the censoring information is the Cox Proportional Hazards (PH) model [26]. When the number of covariates is larger than the number of cases, as it is the case in a typical microarray dataset, the estimates obtained from the Cox model are non-unique and unstable. To cope with the high dimensionality of the microarray dataset, several authors have proposed penalized partial likelihood approaches for the Cox PH model. Li and Luan [66] transformed the Cox partial likelihood using kernels based on a penalization method. However, Engler and Li [36] pointed out that Li and Luan's approach does not show how to select the genes to be included in the prediction of the survival function. Gui and Li [44] proposed a penalized method for Cox regression based on Least Angle Regression (LARS) algorithm of Efron [35]. However, when the penalty function is not strictly convex, as in LARS, and provided that the covariates are highly correlated, Gui and Li's approach often identifies only one of the covariates and ignores the others as pointed out by Engler and Li [36]. Gui and Li [43] also proposed to estimate the regression parameters using a threshold gradient descent minimization of the Cox partial likelihood, but in their approach, the number of selected genes is sensitive to small changes in the threshold parameter (Engler and Li [36]).

Another approach to deal with the high dimensionality of the covariate space is to use a two-stage procedure. The dimension of the original data matrix is reduced from N x p to N x k, where $k < N$, using dimension reduction techniques in the first stage,

and the regression model is used with the reduced data set in the second stage. The performance of the different dimension reduction methods employing the two-stage procedure was investigated extensively in the literature using the Cox model and the Accelerated Failure Time (AFT) model in the second stage. Both the Cox and AFT models are discussed in the Simulation Results chapter 3. For the Cox model, Nguyen and Rocke [78, 79, 80] concluded that Partial Least Squares (PLS), and modified versions of PLS (MPLS) which incorporate the censoring, outperform Principal Component Analysis (PCA) in terms of classification accuracy and mean squared error of the estimated survival function. Bura and Pfeiffer [21] concluded that Sliced Average Variance Estimation (SAVE) is better than Sliced Inverse Regression (SIR) in terms of classification accuracy of tumor classes. Boulesteix and Strimmer [17] combined Partial Least Squares (PLS) with Linear Discriminant Analysis (LDA). The approach outperforms several classification methods such as Nearest Neighbor (NN), Prediction Analysis of Microarray (PAM) and Support Vector Machines (SVM). Bair et. al. [9] stated that Supervised Principal Component Regression (SPCR) outperforms both PCA and PLS in terms of classification error of tumor subtypes. Dai et. al. [27] concluded that PLS and Sliced Inverse Regression (SIR) outperform PCA in terms of classification error rates. Bovelstad et al. [18] stated that PCA performed slightly better than SPCR in terms of the log-rank test, prognostic index and the deviance in the Cox model. In Zhao and Sun [100], Correlation Principal Component Regression (CPCR) is as competitive as modified versions of PLS in terms of root mean squared error of prediction of martingale residuals in the Cox model, and in terms of classification accuracy.

For the AFT model, Huang and Harrington [49] combined PLS and PCA with the Buckley-James algorithm [20] and Leurgens method [65] to handle right-censored

data. Datta et al. [30] combined PLS with three nonparametric approaches to incorporate right-censored data: reweighting, mean imputation and multiple imputation. It turns out that both imputation approaches outperform the reweighting approach in terms of mean squared error of prediction. However, none of the methods presented in the literature perform better than all the other methods under both the Cox and AFT models.

This work focuses on two dimension reduction methods. We first focus on the dimension reduction method of Random Projection (RP), and its motivation, the Johnson-Lindenstrauss (JL) Lemma. The JL Lemma concerns with the projection of points from high-dimensional space to low-dimensional space using the criterion of preserving pairwise distances among the points with a small distortion as opposed to an optimization criterion as in the case of methods such as Principal Component Analysis (PCA) and Partial Least Squares (PLS). In this work, the JL Lemma is revisited when the random projection matrices are Gaussian (entries of the random matrices are independent and identically distributed (i.i.d.) standard Gaussians) or of Achlioptas type (entries of the random matrices are i.i.d. ± 1 with probability $1/2$, or $\pm\sqrt{3}$ with probability $1/6$ and 0 with probability $2/3$). Since the random projection matrix is of dimension p by k, it is important to obtain a small value for k. This work focuses on improving the lower bound for k. For the Gaussian random matrix, an improvement is provided for the lower bound for k obtained from the JL Lemma using pairwise L_2 distances in the space of the original points and pairwise L_2 distances in the space of the projected points (L_2-L_2). An improvement for the lower bound for k using L_2-L_1 distances is also provided. For the Achlioptas-typed random matrix, an alternate proof of the Achlioptas Lemma is provided, and an improvement on the Achlioptas bound using L_2-L_2 distance is obtained, and a lower bound for k

using L_2-L_1 distance is presented.

Also in this work, a variant of PLS, denoted by Rank-based Modified Partial Least Squares (RMPLS), is proposed. The method is insensitive to outlying values in both the predictors and response, and also incorporates the censoring information. The weight vectors for RMPLS can be derived as solution to an optimization problem. Simulation results as well as results for real datasets under the Cox and AFT models indicate that RMPLS works well when outliers are present in the response, and is competitive with other leading methods including PLS in the absence of outliers (see Nguyen and Rojo [81, 82] for details).

This work is organized as follows. Chapter 1 describes DNA microarray technology, and its numerous applications and challenges. Chapter 2 provides a literature review on several well-known dimension reduction methods, including PCA and PLS. Chapter 3 describes the method of Random Projection and the Johnson-Lindenstrauss (JL) Lemma. An improvement to the JL bound for k using L_2-L_2 distance, and an improvement on the lower bound for k using L_2-L_1 distance are provided. Chapter 4 presents the method of Rank-based PLS (RMPLS). The derivation of the weight vectors for RMPLS is provided. The performance of our proposed method RMPLS is assessed by comparing it with several other dimension reduction techniques via a simulation study and real microarray datasets under the Cox and AFT models. Discussions and conclusions are given in chapter 5. Proofs of relevant theorems, plots and tables are provided in the Appendix. For completeness, relevant materials are also presented in the Appendix. Examples include the derivations of the weight vectors for PCA and PLS, algorithm for Sliced Inverse Regression (SIR), the Kaplan-Meier and Nelson-Aalen estimators for the survival function, and the distribution of the errors in the Accelerated Failure Time (AFT) model.

Chapter 1

DNA Microarray

Traditional methods in molecular biology usually work on a "one gene one experiment" basis, which limits the throughput and the understanding of gene functions (Shi [92]). A relatively new technology, called DNA microarray, allows researchers to monitor thousands of genes simultaneously in a single experiment, and thus, helps researchers have a better understanding of the interactions among the genes. There are two major applications for DNA microarray technology: the first is the identification of gene sequences (for example, identification of sequence changes in a genetic mutation), and the second is the determination of expression level or abundance of genes (Shi [92]). Therefore, microarray technology definitely has an impact in many fields such as cancer and toxicological research, and drug study and design. However, as data from microarray experiments accumulates, it is essential to develop better statistical methods and models for their analysis.

This chapter describes the DNA Microarray technology. In particular, two such technologies, the oligonucleotide microarray and the cDNA microarray, are discussed. Applications and challenges of the microarray technology are provided.

1.1 DNA Microarray

A DNA microarray consists of an orderly arrangement of DNA fragments representing the genes of an organism (Coe and Antler [25]). A microarray experiment is often

created by use of robotics to deposit the sample on microplates or standard blotting membranes. Microarrays often contain over 30,000 sample probes or spots, and these probes are typically less than 200 microns in diameter (Coe and Antler [25], Shi [92]). The underlying principle of DNA microarray is *hybridization* or base-pairing, in which DNA nucleotide bases will hybridize with certain other DNA nucleotide bases. When the DNA microarray is immersed in a cellular sample, the messenger RNA (mRNA) within the cells, which is important in the process of protein synthesis, will hybridize to complementary strands of DNA contained in the microarray (Kharal [58]). Using fluorescent labeling, the labeled mRNA that hybridizes with DNA fragments on the microarray will be identifiable as glowing spots on the microarray, while the mRNA that does not hybridize will be invisible (Coe and Antler [25]). Thus, microarrays are used to measure the mRNA expression.

Two main types of microarray technology are predominant: oligonucleotide (Affymetrix) and cDNA microarrays. The goal of any microarray technology is to derive an expression level, quantified by a scalar value, of each gene. High expression levels indicate high amount of genetic activity for a gene, and low expression levels indicate low genetic activity for a gene (Kharal [58]). Each microarray technology uses the same principle of measuring the presence of mRNA contained in the cells of the sample. The oligonucleotide microarrays and cDNA microarrays are discussed next, and the applications and challenges of microarrays will be presented in detail in the following subsections.

1.1.1 Oligonucleotide Arrays

Oligonucleotide arrays are trademarked as GeneChip by Affymetrix (Coe and Antler [25]). The probe or spot location is indicative of the gene identity on the GeneChip.

Within each probe, there are millions of copies of the DNA fragment, and there are up to 20 probe pairs for each gene (Dudoit et al. [34]). The probes are selected so that the *cross-hybridization* with other DNA fragments is minimized. One way to achieve this is to pair multiple probes that work (meaning that the nucleotide bases hybridize correctly) with those that do not work correctly. This setup is known as *Perfect Match (PM) and Mismatch (MM)* probe-pairing. The MM probe is identical to the PM probe except in a single mismatch in the central position of the *oligo* (short DNA subsequence) (Harr and Schlotterer [45]), and serves as a measure of the degree of cross-hybridization or non-specific binding of the mRNA. An example of PM/MM design is shown below:

Reference sequence: *AATGGGTCAGAA***G***GACTCCTATGT*

PM: *TTACCCAGTCTT***C***CTGAGGATACA*

MM: *TTACCCAGTCTT***G***CTGAGGATACA*

The microarray is immersed in a cellular sample, from which the fluorescently labeled mRNA will hybridize to the DNA fragments on the array. The mRNA is not measured directly from the sample. Rather, the mRNA is initially fluorescently labelled a specific color so that if hybridization occurs, a glowing spot on the array is seen. If hybridization does not occur, then the mRNA washes off the array slide. The level of *genetic expression* is assessed by the amount of mRNA produced. A bright glowing spot on the array indicates an *abundant* expression level, and black or light color in the array spot indicates an *inactive* gene in the sample (Kharal [58]). Since the fluorescence strength or intensity in each probe indicates the amount of genetic expression, this intensity measure is scanned and transformed into a numerical value. Since each gene is represented by more than one probes, the expression level for each

gene is its total expression levels across all the probes (Kharal [58]). A typical picture of an oligonucleotide microarray is shown in Figure 1.1.

1.1.2 cDNA Arrays

A cDNA array is a different technology from oligonucleotide array, but the same principle of hybridization is employed in both technologies. In cDNA arrays, the probes are larger pieces of DNA that are complementary to the genes of interest (Coe and Antler [25]). An experiment using cDNA array involves preparing two samples: one is a control sample and the other is an experimental sample, as illustrated in Figure 1.2. The mRNA is extracted from both samples, control and experimental, and is converted into *complementary DNA (cDNA)*. The cDNA is labeled with a fluorescent dye. The two samples are then mixed together and hybridized to the array, and the differences in gene expression levels are revealed by the fluorescent patterns on the array (Coe and Antler [25]). For example, a green fluorescent dye can be used to label the control sample, and a red fluorescent dye to label the experimental sample. Since the samples are mixed together, they would compete against one another in binding to the probes on the array, in which the sample containing more gene expression for a particular probe will win out. So, if there is more mRNA in the control than in the experimental sample, then there will be more mRNA in the control binding to the array, and thus, the probe on the array will fluoresce green. If there is more mRNA in the experimental sample than in the control, the reverse is observed, and the probe on the array will fluoresce red. If there is the same amount of mRNA transcripts in both samples, then the dyes will cancel each other out, and the probe will fluoresce yellow.

Figure 1.1 : Oligonucleotide Microarray. This figure was taken from [98]

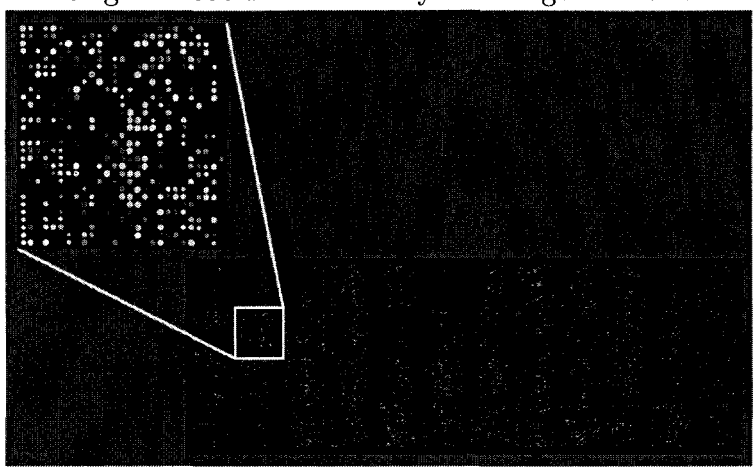
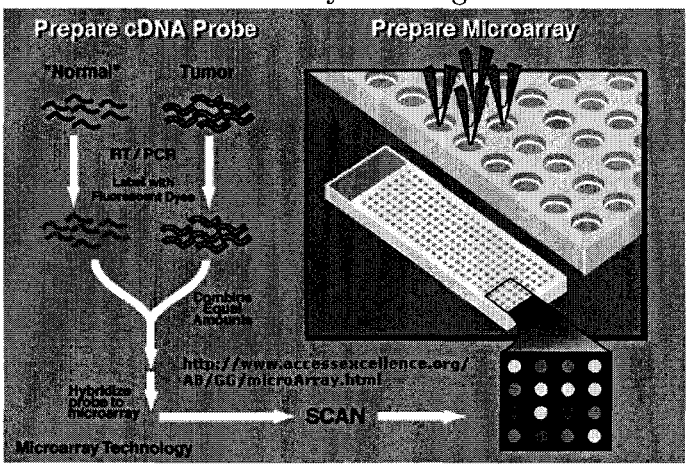


Figure 1.2 : cDNA Microarray. This figure was taken from [2]



1.1.3 Applications and Challenges of Microarrays

Because microarrays contain samples of a large number of genes, they allow researchers to monitor thousands of genes simultaneously in a single experiment. Thus, they can be used to study gene expression levels in a single sample, or to compare gene expression levels in two different samples such as in comparing healthy and diseased tissue samples. Also, researchers are able to study the functions of new genes based

on similarities in the expression patterns with those of known genes. Furthermore, microarrays allow researchers to study the inter-relationships among the genes, and aid in the identification of the gene involved in the development of various diseases (ncbi website [77]). Identification of a diseased gene is beneficial since researchers can target them for therapy, determine a person's risk of developing the disease, and gain insight into the seriousness of the disease (Coe and Antler [25]). For instance, in cancer research, microarrays allow the rapid identification of which genes are turned on and off in tumor development, and thus, researchers can target these genes for therapy. Therefore, microarray technology is applicable to many fields such as cancer and toxicological research, gene therapy, and drug study and design.

There are unique challenges that microarrays pose for researchers despite their great benefits. A typical microarray dataset contains thousands of covariates corresponding to the expression of the genes, which far exceed the number of cases, which is only in the order of hundreds. Existing statistical methods such as the commonly used linear regression model and survival analysis require less covariates than cases. Also, having so many covariates relative to so few samples (cases) creates a high likelihood of finding *false positives* that are due to chance - both in finding differentially expressed genes, and in building predictive models (Piatetsky-Shapiro and Tamayo [86]). The large microarray datasets mandate the application of sophisticated computer algorithms and invite numerous views on the interpretations of biological meaning (Petricoin et al. [85]). Furthermore, the gene expressions are often highly correlated, which makes the analysis even more difficult. Also, there is a lack of evidence supporting the reproducibility, reliability, precision and accuracy of data derived from global gene expression technologies applied across platforms to identical samples (Petricoin et al. [85]). Furthermore, the variation in microarray technology,

such as the fact that the dye balance, with probe intensity and with spatial position on the array, needs to be adjusted or *normalized* (Smyth and Speed [93]).

As mentioned in the Introduction, one approach to cope with the high dimensionality of the microarray dataset is to employ a two-stage procedure: first apply dimension reduction methods to the microarray dataset to obtain a reduced data matrix, and then apply regression models to the reduced data matrix. A literature review of the leading dimension reduction methods is provided in the next chapter. The regression models that handle the censoring information such as the Cox Proportional Hazards and the Accelerated Failure Time models are discussed in chapter 4.

Chapter 2

Literature Review of Dimension Reduction Methods

Due to the high dimensionality of the microarray data, one needs to employ dimension reduction methods to the data matrix before carrying out any statistical analysis. This work focuses on linear dimension reduction methods. This chapter describes several leading methods in the literature: Principal Component Analysis (PCA), Partial Least Squares (PLS), modified versions of PLS that incorporate the censoring information, Univariate Selection (UNIV), Supervised Principal Component Regression (SPCR), Correlation Principal Component Regression (CPCR), and Sliced Inverse Regression (SIR). Chapter 3 discusses the dimension reduction method of Random Projection (RP), and its motivation, the Johnson-Lindenstrauss (JL) theorem. Improvements to the bounds of the reduced dimension obtained from the JL Lemma are provided in this section. Chapter 4 presents a variant of PLS, denoted by Rank-based Modified PLS (RMPLS), that is insensitive to outlying observations in both the response and the covariates. Also, the derivation of the weight vectors of RMPLS as solution to an optimization problem is provided. Results from a simulation study and real datasets indicate that RMPLS works well in the presence of outliers in the response and is comparable to MPLS and PCA in the absence of outliers.

We begin this chapter with the goals of using dimension reduction methods.

2.1 Goals of Dimension Reduction

The goal of reducing the high dimensionality of microarray data is to transform the large number (p) of original gene expression levels to a smaller number ($k \ll n \ll p$) of linear combinations of gene expression levels. Linear dimension reduction methods involve creating a set of orthogonal linear combinations of the original data and then selecting a subset of these based on some criteria associated with the ability of the elements of this subset to predict the response (Nguyen and Rojo [81]). Dimension reduction is grouped into two strategies: feature selection and feature extraction (Van Wieringen et al. [95]). *Feature selection* selects the best possible subset of the gene expression dataset in order to preserve the interpretability of the original data. *Feature extraction* transforms the high-dimensional original data to a low-dimensional data space such that the new features are a linear or nonlinear transformation of the original features, and this transformation seeks to retain most of the relevant information in the original data. Although feature extraction may improve prediction accuracy, it may lack a clear physical interpretation. Also, feature selection is indeed a special case of feature extraction. Dimension reduction strategies are further characterized by univariate versus multivariate approaches, and supervised versus unsupervised approaches (Van Wieringen et al. [95]). *Univariate approaches* consider each individual gene separately, while *multivariate approaches* consider the correlation among the genes. *Supervised approaches* consider the response (survival) information, while *unsupervised approaches* completely ignore the response in the dimension reduction. This chapter describes several leading methods in the literature that are supervised (for example, PLS), unsupervised (PCA), univariate (UNIV), multivariate (PCA, PLS), feature extraction (PCA, PLS), and feature selection (UNIV).

Next, we set forth the notation used throughout this work.

2.1.1 Notation

Let X be the N x p matrix of centered gene expression values (i.e., the p columns of X are centered by subtracting the corresponding column mean from the column entries), where N is the number of cases (patients), and p is the number of genes and $N \ll p$. Let y be the N x 1 vector of true survival times, c be the N x 1 vector of right-censoring times, and let y and c be independent. The observed data consists of the data matrix X, the survival times $T_i = min(y_i, c_i)$, and censoring indicators $\delta_i = I(y_i \leq c_i)$ for $i = 1, \ldots, N$ ($\delta_i = 0$ if censoring occurs, and $\delta_i = 1$ if the true survival time is observed).

Also, the following notation is adopted throughout this work. For matrix A and column vector a, denote by A^T and a^T the transpose of matrix A and vector a, respectively.

We now describe the dimension reduction techniques.

2.2 Principal Component Analysis (PCA)

PCA is a well-known dimension reduction technique that involves transforming the original high dimensional set of (possibly correlated) gene expression levels to a reduced set of uncorrelated (orthogonal) gene components (principal components). The Principal Components (PCs) can be obtained through the the spectral decomposition of the sample covariance matrix, which equals $S = \frac{1}{N-1} X^T X$ because X is centered. Since S is symmetric, it can be diagonalized by the orthogonal matrix of its eigenvectors,

$$S = V \Delta V^T$$

where the N x N matrix $\Delta = \text{diag}(\lambda_1 \geq \cdots \geq \lambda_N)$ and $(\lambda_k)_{k=1}^N$ represent the eigenvalues of S in descending order, and the columns of the p x N orthogonal matrix $V = (v_1, \ldots, v_N)$ are the corresponding eigenvectors that provide the weights for the linear combinations. Since the weight vectors, $w_k = v_k$, are constructed such that they are unit vectors, $w_k^T w_k = 1$, the proportion of the variation explained by the k^{th} PC is λ_k/p, the cumulative proportion for the first k PCs is $\sum_{i=1}^k \lambda_i/p$, and the total variation explained by all the N PCs is $\sum_{i=1}^N \lambda_i = p$. Since $\lambda_1 \geq \cdots \geq \lambda_N$, and the first few λ's are large, the PCs are linear combinations of the original gene expression levels such that the first few PCs explain most of the variation in the original data.

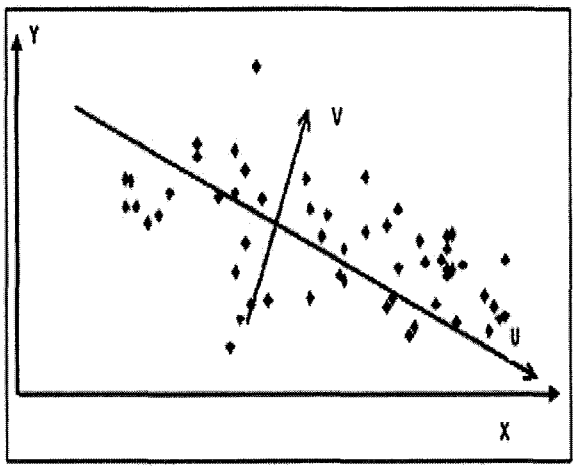
Mathematically, the weight vectors of PCA are constructed sequentially by maximizing the variance of the linear combinations of the gene expression levels (covariates) such that these linear combinations are uncorrelated,

$$w_k = \underset{w^T w=1}{\arg\max}\ \text{Var}(Xw) = \underset{w^T w=1}{\arg\max}(N-1)^{-1} w^T X^T X w$$

subject to the constraint $w_k^T X^T X w_j = 0$ for all $1 \leq j < k$, where $k = 1, \ldots, min(N, p)$. The k^{th} Principal Component (PC) is $\tilde{x}_k = X w_k$. The orthogonal constraint in the optimization criterion ensures that the PC's are orthogonal or uncorrelated, i.e. $Cov(Xw_k, Xw_j) = 0$ for $k \neq j$. Geometrically, the PCs represent a new coordinate system obtained by rotating the original coordinate system, in such a way that the new axes represent the directions of maximum variability in the original data, and are ordered in terms of the amount of variation of the original data they account for (Dai et al. [27]). This is illustrated in Figure 2.1 with data in two dimensions. We should note that the construction of the PC's does not involve the response, and thus, the components with the highest variation explained (the largest eigenvalues λ's) are not necessarily predictive of the response in a multivariate regression model. Details on the method of PCA can be found in Joliffe [53] and Mardia [72]. Detail of the

eigenvalue (spectral) decomposition of PCA is provided in Appendix A.

Figure 2.1 : Principal Component Analysis: a simple example. The data are in two dimensions. The first Principal Component (PC) is labeled U, and the second PC (labeled V) is orthogonal to the first PC. This figure was taken from [76].



The number of PCs, k, is often chosen by cross-validation with a certain optimization criterion such as minimizing the mean squared error of prediction. Another approach is to select the first k PCs that explain a certain percentage of total variation explained in the original data. This approach relies on the fact that the first few PCs capture most of the variation explained in the original data, and thus, the rest of the PCs can be ignored without losing much of the information contained in the original data. Other approaches include the Kaiser's criterion [55] which excludes the PCs whose eigenvalues are less than the average, and Cattell's scree graph [23] which examines the percentage of variation explained by each PC. The latter approach is a good visual tool to assess the contribution in terms of variation explained of each PC.

2.3 Univariate Selection (UNIV)

The method of Univariate Selection (UNIV) fits a univariate regression model of y against each of the genes, and obtains a $p-$value from the test of the null hypothesis $\beta_j = 0$ versus the alternative $\beta_j \neq 0$ (Bovelstad et al. [18]). The genes are then ranked according to increasing $p-$values, and the top-ranked k genes are selected, where k is either fixed or selected by cross validation. This work uses the Cox and AFT models as the regression model. Unlike PCA, UNIV ignores the correlation among the covariates, which may cause many of the selected covariates to have insignificant $p-$values in the multivariate regression model (Van Wieringen et al. [95]).

2.4 Supervised Principal Component Regression (SPCR)

One major drawback of PCA is that the method completely ignores the response in its construction of the components. Bair and Tibshirani [8, 9] proposed a variant of PCA, which they called Supervised Principal Component Regression (SPCR). This method employs univariate selection (UNIV) to pick out a subset of the original gene expressions that are correlated with the response, and then applies PCA to that subset. One criterion to select the subset of genes is to obtain the λ_{SPCR} percent of the top ranked genes according to the p-values from UNIV.

2.5 Correlation Principal Component Regression (CPCR)

Sun [94] proposed a variant of SPCR, called Correlation Principal Component Regression (CPCR). The first step to CPCR is to do principal component analysis (PCA) on the gene expression data matrix X, but retaining all $k = \min(p, N)$ principal components. In the context of regression, the second step to CPCR involves regressing

the response variable y on each of the k PC's, and select $k_1 < k$ PCs that have the highest correlations with the response y (Sun [94]). Similar to SPCR, CPCR takes into account the response variable, while PCA does not.

The response variable is usually censored, and hence, the correlation between the censored response and the PC's cannot be computed. A variant of CPCR is proposed by Zhao and Sun [100] to incorporate the censoring. The first step involves the construction of all the k PC's, and the second step is to obtain the PC's, denoted by $\tilde{X}^* = (\tilde{x}_1^*, \ldots, \tilde{x}_{k_1}^*)$, ordered from smallest to largest based on the p-values of the coefficients in the Cox or AFT models when regressing the censored y on the PCs individually.

2.6 Sliced Inverse Regression (SIR)

Conventional regression models estimate the forward regression function $E(y|X)$, which is a p-dimensional surface and difficult to estimate when $p \gg N$. Sliced Inverse Regression (SIR), first proposed by Li [67], focuses on the inverse regression function $E(X|y)$, which consists of p one-dimensional regressions, and is easier to estimate. Since the response y (survival times) is continuous, SIR first replaces y by its discrete version, denoted by $\tilde{y}$, which is constructed by slicing the range of y onto H intervals. One way to partition y is by its quantiles, so that the number of cases in each slice is not too small. Within each of the H slices, a p-dimensional vector of the mean of X is obtained, i.e. $\bar{X}|y_h$, for $h = 1, \ldots, H$, and y_h corresponds to the cases of the response y in slice h. The projection vectors v_k are then obtained through the general eigenvalue decomposition of the sample covariance matrix of $\bar{X}|y_h$, denoted by $S_{\bar{X}|y_h}$ where $h = 1, \ldots, H$, with respect to the sample covariance matrix of X, denoted by

S_X. In other words, the eigenvalue decomposition is given as

$$S_{\bar{X}|y_h} v_k = \lambda_k S_X v_k$$

subject to the constraints $v_k S_X v_k = 1$. Here, λ_k is the k^{th} eigenvalue of $S_X^{-1} S_{\bar{X}|y_h}$ in descending order, and the v_k is the corresponding eigenvector.

The k^{th} SIR component is $\tilde{x}_k = X v_k$. SIR does not require any traditional assumption on the distribution of $y|X$, so any model can be applied in the analysis. Also, SIR incorporates the response (survival times) in conjunction with gene expression data (covariates). Details on SIR can be found in Li [67], Li et al. [68], Li and Li [69], and Dai et al. [27].

Since SIR is designed for an uncensored response, the method cannot be applied directly to censored survival data. Li, Wang, and Chen [68] proposed a *double slicing* procedure to bypass this censoring problem. The approach first partitions the response y into a censored part and an uncensored part. The slicing is done within those two parts separately, but the two parts are combined for the final eigenvalue decomposition. Li and Li [69] pointed out that the implementation of SIR requires the sample covariance matrix S_X to be non-singular. However, the gene expression data matrix is of dimension N x p, where $N < p$, which causes S_X to be singular. They propose to first reduce the dimension of p to k, where $k < N \ll p$, via a dimension reduction method such as PCA or PLS, and then apply SIR to these k components. In this work, Li and Li's approach is adopted in the simulations. The algorithm to compute the sample SIR weights with a censored response is provided in Appendix A.

2.7 Partial Least Squares (PLS)

Herman Wold [99] introduced the method of Partial Least Squares (PLS), which gained popularity in the field of econometrics, and later in chemometrics and sensory evaluation (Geladi [40]). The PLS weights are obtained sequentially by maximizing the covariance between the linear combinations of the original covariates X and the response y,

$$w_k = \underset{w^T w=1}{\arg\max}\ \mathrm{Cov}(Xw, y) = \underset{w^T w=1}{\arg\max}(N-1)^{-1} w^T X^T y \tag{2.7.1}$$

subject to the constraint $w_k^T X^T X w_j = 0$ for all $1 \leq j < k$, where $k = 1, \ldots, min(N, p)$, as in PCA. Here, the w_k's are the column vectors of the weight matrix W, and are defined so that the squared sample covariance between the response y and the *scores* components, Xw_k, is maximal under the condition that the *scores* components are mutually uncorrelated (Boulesteix and Strimmer [17]). In other words, PLS seeks directions that have high covariance with the response. Since PLS uses the response y to construct its directions (PLS components), Nguyen and Rocke [79] pointed out that PLS weights are non-linear functions of both the covariates and the response variable, rather than just the covariates as in PCA. Hence, the construction of each of the PLS components takes into account the weight of the covariates on y, i.e. the strength of the covariates' univariate effect on y (Hastie et al. [46]). The k^{th} PLS component is obtained as $\tilde{x}_k = Xw_k$. The derivation of PLS as an eigenvalue problem is provided in Appendix A.

Algorithms to compute the weight vectors, w_k's, are given in De Jong [31], Denham [33], Hoskuldsson [48], and Martens and Naes [73]. A good review of the different methods of PLS is given in Boulesteix and Strimmer [17]. The simulations provided in this work uses the orthogonal scores algorithm of Martens and Naes [73]. The

algorithm is given below:

1. The p columns of X and vector y are standardized (mean 0 and variance 1).
2. Let $\tilde{w} = X^T y$; define the weight vector w as $w = \frac{\tilde{w}}{\sqrt{\tilde{w}^T \tilde{w}}}$.
3. Let $\tilde{t} = Xw$, define the scores vector t as $t = \frac{\tilde{t}}{\tilde{t}^T \tilde{t}}$.
4. Find $q_1 = y^T t$, and $q_2 = X^T t$.
5. Deflate X and y: $X = X - tq_2^T = (I_N - tt^T)X$ and $y = y - tq_1^T = (I_N - tt^T)y$, where I_N is the N x N identity matrix.

The k weight vectors are obtained sequentially by repeating the algorithm.

The PLS objective criterion (2.7.1) takes into account the response variable, while PCA does not. For this reason, PLS is termed a *supervised* method while PCA is an *unsupervised* method. However, PLS does not incorporate the censoring information, which induces bias in the estimates. Improvements to this approach were proposed by combining the construction of PLS components and the Cox regression, and hence, incorporating censoring into the construction of PLS components. Park, Tian and Kohane [84] reformulated the Cox model as a standard Poisson regression model and derived the PLS components from the formulation of PLS for the generalized linear models. The equivalence of the Poisson model and the Cox model was shown in Whitehead [97], and the formulation of the PLS for the generalized linear models was shown in Marx [74]. However, Park's algorithm may fail to converge when the number of covariates is large (Gui and Li [42]). Gui and Li [42] proposed the Partial Cox Regression (PCR), which involves the construction of predictive components by repeated least square fitting of residuals and Cox regression fitting. These components can then be used in the Cox model.

Nguyen and Rocke [79] proposed a modification of the PLS approach, denoted by Modified Partial Least Squares (MPLS), which modifies the PLS weights in the dimension reduction step by use of the Cox regression to incorporate censoring. Datta et. al. [30] applied three nonparametric approaches to incorporate right-censoring in the PLS method in the context of a linear regression model: reweighting, mean imputation and multiple imputation. The approaches of mean imputation and multiple imputation perform relatively the same. Thus, only reweighting and mean imputation are discussed in detail in this work. Nguyen and Rocke's MPLS is discussed in the next subsection.

2.7.1 Modified Partial Least Squares (MPLS)

Nguyen and Rocke showed that the PLS weights (2.7.1) can be expressed as,

$$w_k = \sum_{i=1}^{N} \theta_{ik} v_i$$

where v_i's are the i^{th} eigenvectors of $X^T X$. Closed form expressions for the constants θ_{ik} are given in Nguyen and Rocke [79]. As pointed out by Nguyen and Rocke, the scalars θ_{ik} depend on the response y only through the dot product $a_i = u_i^T y$, where u_i's are the eigenvectors of XX^T [79]. The estimated slope coefficient of the simple linear regression of y on u_i is $y^T u_i/(u_i^T u_i)$, and if the gene expression matrix is centered, then $u_i^T u_i = 1$. Hence, the dot product a_i is also the slope coefficient in the simple regression of y on u_i. Since the response is censored, it is sensible to replace this dot product by the slope coefficient obtained from the univariate Cox regression of y on u_i. When the AFT model is used instead of the Cox model in the second stage, we propose to replace a_i by the slope coefficient obtained from the univariate AFT regression of y on u_i (see Nguyen and Rojo [82] for the details of MPLS under the

AFT model). We denote these methods by Modified Partial Least Squares (MPLS).

2.7.2 Partial Least Squares with Right-Censored Responses in Linear Regression

Datta [30] considered three approaches to handle right-censored responses in the Accelerated Failure Time (AFT) model: reweighting, mean imputation, and multiple imputation. They incorporated these three approaches with PLS. Since the approaches of mean imputation and multiple imputation perform relatively the same, only the reweighting and mean imputation are discussed below.

1) **Reweighting (RMPLS) (or Inverse Probability of Censoring Weighted)**: Assuming that the true survival time y is independent of censoring time c given the covariates, the Kaplan-Meier estimator can be used to estimate $S_c(t)$, the survival function of the censoring time c, as follows

$$\hat{S}_c(t) = \prod_{t_i \leq t} \left[1 - \frac{c_i}{N_i}\right], \tag{2.7.2}$$

where $t_1 < \cdots < t_m$ are the distinct ordered censored times, c_i is the number of censored observations at time t_i, and N_i is the number of individuals at risk prior to time t_i. Under this method, the censored response is replaced with 0, but the uncensored response is reweighted by the inverse of the probability that it corresponds to a censored observation. In other words, let $\tilde{y}_i = 0$ for $\delta_i = 0$ and $\tilde{y}_i = T_i/\hat{S}_c(T_i-)$ for $\delta_i = 1$, where $T_i = min(y_i, c_i)$, and $-$ denotes the left limit. PLS is then used with $\tilde{y}$ and X. This method is denoted by $RWPLS$.

2) **Mean Imputation (MIPLS)**: Under this scheme, the uncensored response T_i is kept, but the censored T_i is replaced by its expected value given that the true

survival time y_i exceeding the censoring time c_i. This conditional expectation can be estimated by the Kaplan-Meier curve,

$$y_i^* = \frac{\sum_{t_j > c_i} t_j \Delta \hat{S}(t_j)}{\hat{S}(c_i)}$$

where t_j are the ordered death times, $\Delta \hat{S}(t_j)$ is the jump size of $\hat{S}$ at t_j, and $\hat{S}$ is the Kaplan-Meier estimator of the survival function of y with the roles of δ and $1-\delta$ switched in Eq. (2.7.2). Under this method, we let $\tilde{y}_i = y_i$ if $\delta_i = 1$ and $\tilde{y}_i = y_i^*$ if $\delta_i = 0$. As in the case of reweighted PLS, the usual PLS method is used with $\tilde{y}$ and X. This approach is denoted by $MIPLS$.

In the next chapter, the dimension reduction of Random Projection (RP), and its motivation, the Johnson-Lindenstrauss (JL) Lemma are discussed. Improvements to the lower bound for k obtained from various versions of the JL Lemma are provided in this chapter.

Chapter 3

Random Projection

3.1 Introduction

Among the various dimension reduction methods discussed in the literature, Random Projection (RP) has attracted a lot of attention lately. RP is a computationally-simple method of dimension reduction whereby the original p-dimensional data matrix X is projected onto a k-dimensional subspace by multiplying the N x p data matrix X by a p x k random projection matrix Γ. In matrix notation,

$$\tilde{X} = X\Gamma$$

where X is N x p data matrix, Γ is a p x k random projection matrix, and $\tilde{X}$ is the resulting N x k matrix consisting of the projected points onto a lower $k-$dimensional subspace. Orthogonality of the projection matrix preserves similarities, e.g. the inner product or the Euclidean distance, of the original vectors when projected to the low-dimensional space. Although the random matrix Γ is not orthogonal, the loss of information is minimal because the orthogonality property is achieved with high probability in high-dimensional space (Achlioptas [3], Goel et al. [41], Hecht-Nielsen [47]).

Random Projection methods (RP) have been used in numerous areas of research. In the area of nearest-neighbor queries, Kleiberg [60] developed a new approach to the nearest-neighbor problem by combining randomly chosen one-dimensional projections of the underlying data. Indyk and Motwani [50] used RP to solve the nearest-neighbor

problem in high dimensions. In the area of machine learning, Arriaga and Vempala [7] used RP to build a model of robust concept learning. Dasgupta [29] combined RP with Expectation-Maximization algorithm to cluster Gaussian mixture models in high dimensions. Fern and Brodley [37] used RP in a cluster ensemble approach. Candes and Tao [22] used RP to recover discrete signals as sparse superposition of sinusoids. Deegalla and Bostrum [32] combined RP with the nearest-neighbor classifier for image and microarray data. In the area of organizing text and audio documents and image data, Kaski [57] and Kohonen et al. [61] combined RP with self-organizing maps (SOM) to organize text documents. Papadimitriou et al. [83] combined RP with latent semantic indexing (LSI) to classify documents. Kurimo [62], in a similar approach, combined RP with LSI to index audio documents. Bingham and Manilla [16] applied RP in the processing of images, and information retrieval in text documents. In the area of object and face recognition, Goel et al. [41] applied RP to face recognition experiments. Li et al. [71] combined RP with EM algorithm to classify objects based on their geometric appearance. In the area of gene expression clustering, Bertoni and Valentini [12, 13, 14] combined RP with clustering algorithms to cluster gene expression data.

A good overview on the use of RP is given in Bingham and Mannila [16], and Goel et al. [41]. Unlike several other dimension reduction methods, RP does not obtain a low-dimensional subspace using a certain optimization criteria. For example, Principal Component Analysis (PCA) finds the set of directions sequentially by maximizing the variance of the linear combination of the covariates such that these linear combinations are orthogonal. Furthermore, the performance of RP is comparable to PCA in terms of the average difference of the pairwise Euclidean distances among the points in the projected space and the pairwise Euclidean distances among the points

in the original space for text and image data (Bingham and Mannila [16]), in terms of clustering accuracy for learning mixture of Gaussians (Dasgupta [29]), in terms of classification accuracy for machine learning experiments (Fradkin and Madigan [38]), in terms of nearest neighbor classification accuracy for image data and microarray data (Deegalla and Bostrum [32]), and in terms of recognition accuracy for face recognition experiments (Goel et al. [41]). Moreover, RP is faster to compute than PCA when the dimension of the data is high since the eigenvalue decomposition of the covariance data matrix in PCA is computationally expensive. The computing cost for PCA is $O(Np^2) + O(p^3)$, while that of RP is $O(k^2p)$ when the entries to the random matrix are independent and identically distributed (i.i.d.) standard Gaussians, and $O(kp)$ when the entries are of Achlioptas type (3.1.1) (Bingham and Mannila [16], Goel et al. [41], Li et al. [70]).

The main motivation for Random Projection (RP) is the Johnson-Lindenstrauss Lemma (1984), which states that a set of N points in $p-$dimensional Euclidean space can be mapped down onto a $k = O(\ln N/\epsilon^2)$ dimensional Euclidean space such that the pairwise distance between any two points is preserved within a factor of $(1 \pm \epsilon)$ for any $0 < \epsilon < 1$. The distance measure used in the JL Lemma is the Euclidean distance. In the original proof of the JL Lemma, Johnson and Lindenstrauss [54] show that such a mapping is provided by a random orthogonal projection. However, the form of the random projection matrix is not specified. Frankl and Maehara [39] simplified the original proof of Johnson and Lindenstrauss using geometric techniques, and provided an improvement on the lower bound for k, i.e. $k \geq \lceil \frac{27 \ln N}{3\epsilon^2 - 2\epsilon^3} \rceil + 1$. Indyk and Motwani [50] simplified the proof of the JL Lemma using i.i.d. standard Gaussian entries for the random matrix Γ. Also, using a Gaussian random matrix, Dasgupta and Gupta [28] further simplified the proof with elementary probabilistic techniques

based on moment generating functions, and improved on the lower bound for k to be $k \geq \frac{24 \ln N}{3\epsilon^2 - 2\epsilon^3}$.

Instead of improving the lower bound for k, several papers in the literature focus on improving the computational time of the Random Projections. Achlioptas [3] proposed two simpler distributions for the entries of the random projection matrix Γ as alternatives to using the standard Gaussian distribution:

$$r_{ij} = \begin{cases} +1 & \text{with prob. } 1/2 \\ -1 & \text{with prob. } 1/2 \end{cases} \tag{3.1.1}$$

or

$$r_{ij} = \sqrt{3} \begin{cases} +1 & \text{with prob. } 1/6 \\ 0 & \text{with prob. } 2/3 \\ -1 & \text{with prob. } 1/6 \end{cases} \tag{3.1.2}$$

Using a random projection matrix Γ consisting of entries r_{ij}'s drawn from distribution given in (3.1.2), some sparsity is attained since most of the entries of Γ are 0. An advantage of using the distributions given in (3.1.1) and (3.1.2) as entries to Γ over the choice when the entries are standard Gaussians is in the computational savings. The entries r_{ij}'s of the random matrix defined through (3.1.1) and (3.1.2) can be generalized as follows:

$$r_{ij} = \sqrt{q} \begin{cases} +1 & \text{with prob. } \frac{1}{2q} \\ 0 & \text{with prob. } 1 - \frac{1}{q} \\ -1 & \text{with prob. } \frac{1}{2q} . \end{cases} \tag{3.1.3}$$

Thus, $q = 1$ yields (3.1.1), and $q = 3$ yields (3.1.2). Furthermore, using $q \gg 3$ (e.g. $q = \sqrt{p}$ or $q = \frac{p}{\ln p}$) can significantly speed up the computation (Li et al. [70]) since the

random matrix Γ is *very sparse*. Arriaga and Vempala [7] obtained a slightly worse bound than that of Dasgupta and Gupta with either a Gaussian random matrix or a random matrix consisting of entries drawn from the $Uniform(-1, 1)$ distribution. Ailon and Chazelle [4] extended the idea of using sparse random matrices with a randomized Fourier transform to speed up the RP. Ailon and Liberty [5, 6] improved on the algorithm of Ailon and Chazelle by combining randomized block diagonal matrix with a 4-wise independent deterministic code matrix, and combining tensor products and Lean Walsh Transform with any deterministic matrix. Matousek [75] provided a version of the JL Lemma that allows the entries of the random matrix Γ to be arbitrary independent random variables with zero mean, unit variance and subgaussian tail (see Matousek [75] for a discussion on the variants of the JL Lemma). All these improvements on the time needed to obtain the random projection, however, do not improve on the lower bound for k.

We adopt the following notation to use throughout this work. Denote by $\phi(.)$ and $\Phi(.)$ the standard Gaussian density and cumulative distribution functions, respectively. Denote by L_2-L_2 RP the random projection that uses L_2 distances in the space of points to be projected and L_2 distances in the space of the projected points, and L_2-L_1 RP the random projection that uses L_2 distances in the space of points to be projected and L_1 distances in the space of the projected points. For $\mathbf{x} \in \mathbf{R}^p$, let $||\mathbf{x}||_1 = \sum_{i=1}^{p} |x_i|$, and $||\mathbf{x}||^2 = \sum_{i=1}^{p} x_i^2$.

The JL Lemma allows for the projection of N points in $p-$dimensional Euclidean space onto a $k-$ dimensional Euclidean space, with $k \geq \frac{24 \ln N}{3\epsilon^2 - 2\epsilon^3}$, so that the pairwise distances are preserved within a factor of $1 \pm \epsilon$ with high probability. Note that the JL Lemma deals with the L_2-L_2 Random Projection (RP). By working directly with the distributions of the random distances rather than resorting to the moment

generating function technique, an improvement on the lower bound for k is obtained. The additional reduction in dimension when compared to the bounds found in the literature, is at least 11%, and, in some cases, up to 34% additional reduction is achieved. Using the moment generating function technique, we further provide a lower bound for k for the L_2-L_1 RP. Comparison with the results obtained in the literature shows that the bound presented here provides an additional $36 - 40\%$ reduction.

In subsection 3.2, we describe the JL Lemma, and sketch the proof of Dasgupta and Gupta version of the JL Lemma. Subsection 3.3 provides improvements on the Dasgupta and Gupta lower bound for k by 1) using the moment generating function technique and 2) working directly with the distribution function of the random Euclidean distances. Subsection 3.4 describes the JL Lemma for the L_2 norm using Achlioptas-typed random matrices. Subsection 3.5 provides an alternate proof to the Achlioptas Theorem. Also, an improvement to the Achlioptas bound for the Rademacher random matrices is provided using the properties of the Rademacher random variable. In particular, we improve on the Achlioptas bound by using 1) Hoeffding's Inequality, 2) Berry-Esseen Theorem, and 3) Pinelis Inequality. We further discuss the case for the asymmetric simple random matrices. Subsection 3.6 discusses the L_1-L_1 random projection. Subsection 3.7 provides an improvement on the lower bound for k using the L_2-L_1 random projection with 1) Gaussian random matrices, and 2) Achlioptas-typed random matrices. Discussions are provided in section 3.8.

Next, we discuss the Johnson-Lindenstrauss (JL) Lemma, and the Dasgupta and Gupta version of the JL Lemma from which the lower bound for k is obtained using the moment generating function (mgf) approach.

3.2 Johnson-Lindenstrauss Lemma (L_2-L_2 RP)

In their pioneering work, Johnson and Lindenstrauss [54] provided the following result:

Johnson-Lindenstrauss (JL) Lemma *For any $0 < \epsilon < 1$ and integer N, let k be such that $k = O(\ln N/\epsilon^2)$. For any set V of N points in $\mathbf{R}^p$, there is a linear map $f\colon \mathbf{R}^p \to \mathbf{R}^k$ such that for any $\mathbf{u}, \mathbf{v} \in V$,*

$$(1-\epsilon)\,||\mathbf{u}-\mathbf{v}||^2 \leq ||\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})||^2 \leq (1+\epsilon)\,||\mathbf{u}-\mathbf{v}||^2 \ . \tag{3.2.1}$$

Using a linear map f that is a random orthogonal projection, Johnson and Lindenstrauss [54] showed that with high probability, the event in (3.2.1) is obtained. However, an explicit construction of f is not provided, i.e. the form of the random projection matrix Γ is not specified. Indyk and Motwani [50] and Dasgupta and Gupta [28] gave an explicit form of the mapping f in their versions of the JL Lemma. The mapping is provided by $f(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{x}\Gamma$, where the entries of the random matrix Γ are independent and identically distributed (i.i.d.) standard Gaussians, and $\mathbf{x} \in V$. In a remarkable paper using only elementary probabilistic techniques, Dasgupta and Gupta [28] improved on the lower bound for k from the original JL Lemma as follows.

Dasgupta and Gupta version of the JL Lemma: *For any $0 < \epsilon < 1$ and integer N, let k be such that*

$$k \geq \frac{24 \ln N}{3\epsilon^2 - 2\epsilon^3} \ .$$

Then for any set V of N points in $\mathbf{R}^p$, there is a linear map $f\colon \mathbf{R}^p \to \mathbf{R}^k$ such that for any $\mathbf{u}, \mathbf{v} \in V$,

$$P\left[(1-\epsilon)\,||\mathbf{u}-\mathbf{v}||^2 \leq ||\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})||^2 \leq (1+\epsilon)\,||\mathbf{u}-\mathbf{v}||^2\right] \geq 1 - \frac{2}{N^2} \ . \tag{3.2.2}$$

Let $\mathbf{x} = \mathbf{u} - \mathbf{v}$. Since f is linear, the inequality in (3.2.2) is equivalent to

$$P\left[||\mathbf{f}(\mathbf{x})||^2 \geq (1+\epsilon)\,||\mathbf{x}||^2\right] + P\left[||\mathbf{f}(\mathbf{x})||^2 \leq (1-\epsilon)\,||\mathbf{x}||^2\right] \leq \frac{2}{N^2} \ . \tag{3.2.3}$$

The bound in (3.2.3) can be obtained by separately bounding the left- and right-tail probabilities. That is, by finding f so that simultaneously,

$$P\left[||\mathbf{f}(\mathbf{x})||^2 \geq (1+\epsilon)\,||\mathbf{x}||^2\right] \leq \frac{1}{N^2}\ ,$$

and

$$P\left[||\mathbf{f}(\mathbf{x})||^2 \leq (1-\epsilon)\,||\mathbf{x}||^2\right] \leq \frac{1}{N^2}\ .$$

The proof of Dasgupta and Gupta's version of the JL Lemma hinges on the use of standard Gaussians as entries to the random matrix Γ, and the moment generating function technique. The proof is sketched next, as this will set down the notation and facilitate the reading in subsequent sections.

Sketch of the proof of Dasgupta and Gupta's version of the JL Lemma

Let Γ be a random matrix of dimension p x k with entries $r_{ij} \sim N(0,1)$ independent. For $\mathbf{x} \in V$, define $f(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{x}\Gamma$, and $\mathbf{y} = \sqrt{k}\frac{f(\mathbf{x})}{||\mathbf{x}||}$. Then $y_j = \frac{\mathbf{x}\mathbf{r}_j}{||\mathbf{x}||} \sim N(0,1)$ and $y_j^2 \sim \chi_1^2$ with $E(||\mathbf{y}||^2) = k$, where r_j is the j^{th} column of Γ.

Let $\alpha_1 = k(1+\epsilon)$, and $\alpha_2 = k(1-\epsilon)$. Then the right-tail probability is bounded by

$$\begin{aligned}
P\left[||\mathbf{f}(\mathbf{x})||^2 \geq (1+\epsilon)\,||\mathbf{x}||^2\right] &= P\left[||\mathbf{y}||^2 \geq \alpha_1\right] \\
&\leq E\left[e^{s||\mathbf{y}||^2}e^{-s\alpha_1}\right] \qquad \text{(Markov's Inequality, } s>0) \\
&= \left(e^{-s(1+\epsilon)}E(e^{sy_j^2})\right)^k \qquad \text{(i.i.d. } \mathrm{y}_j) \\
&= e^{-s\alpha_1}(1-2s)^{-k/2}\ , \qquad s \in (0,1/2)\ . \qquad (3.2.4)
\end{aligned}$$

Similarly, the left-tail probability is bounded by

$$\begin{aligned} P\left[||\mathbf{f}(\mathbf{x})||^2 \leq (1-\epsilon)\,||\mathbf{x}||^2\right] &= P[||\mathbf{y}||^2 \leq \alpha_2] \\ &\leq \left(e^{s(1-\epsilon)}E(e^{-sy_j^2})\right)^k , \qquad s>0 \\ &= e^{s\alpha_2}\,(1+2s)^{-k/2} \\ &\leq e^{-s\alpha_1}(1-2s)^{-k/2} , \qquad s\in(0,1/2) , \quad (3.2.5) \end{aligned}$$

where the inequality in (3.2.5) follows from the fact that $e^s/(1+2s)$ is decreasing in $s \in \left(-\frac{1}{2}, \frac{1}{2}\right)$, and hence $\frac{e^{s(1-\epsilon)}}{1+2s} \leq \frac{e^{-s(1+\epsilon)}}{1-2s}$ for $s \in (0, 1/2)$. Thus, the bound for the left-tail probability is the same as that for the right-tail probability. The tightest bound in (3.2.4), and hence in (3.2.5) also, is obtained by minimizing with respect to s. The minimizing $s^* = \frac{1}{2}\left(\frac{\epsilon}{1+\epsilon}\right) \in (0, 1/2)$. Since $g(s) = e^{-s(1+\epsilon)}(1-2s)^{-1/2}$, $s \in (0, 1/2)$, is strictly convex, s^* is the unique minimizer of (3.2.4). Plugging s^* back into (3.2.4) yields

$$\begin{aligned} P[||\mathbf{y}||^2 \geq \alpha_1] &\leq exp\left(-\frac{k}{2}\left(\epsilon - \ln(1+\epsilon)\right)\right) \\ &\leq exp\left(-\frac{k}{12}\left(3\epsilon^2 - 2\epsilon^3\right)\right) , \qquad (3.2.6) \end{aligned}$$

where (3.2.6) is obtained after using the inequality $\ln(1+\epsilon) \leq \epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3}$.

The same bound is obtained for the left-tail probability. Thus, when $k \geq \frac{24 \ln N}{3\epsilon^2 - 2\epsilon^3}$, then both $P\left[||\mathbf{f}(\mathbf{x})||^2 \geq (1+\epsilon)\,||\mathbf{x}||^2\right]$ and $P\left[||\mathbf{f}(\mathbf{x})||^2 \leq (1-\epsilon)\,||\mathbf{x}||^2\right]$ are bounded above by $1/N^2$.

The results are given in terms of the probability that the distance between one pair of points is not substantially distorted when projected, and a lower bound on this probability was chosen as $1 - 2/N^2$. However, in most applications, the user is interested in simultaneously preserving distances among all $\binom{N}{2}$ pairs of distinct points selected from V. Thus, of interest is a lower bound on the probability of the

event

$$\left\{ \bigcap_{\substack{\mathbf{u},\mathbf{v}\in V \\ \mathbf{u}\neq\mathbf{v}}} (1-\epsilon)\left\|\mathbf{u}-\mathbf{v}\right\|^2 \leq \left\|\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})\right\|^2 \leq (1+\epsilon)\left\|\mathbf{u}-\mathbf{v}\right\|^2 \right\}. \tag{3.2.7}$$

Since the probability of this event is bounded below by

$$1-\sum_{\substack{\mathbf{u},\mathbf{v}\in V \\ \mathbf{u}\neq\mathbf{v}}} P\left[\left\{(1-\epsilon)\left\|\mathbf{u}-\mathbf{v}\right\|^2 \leq \left\|\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})\right\|^2 \leq (1+\epsilon)\left\|\mathbf{u}-\mathbf{v}\right\|^2 \right\}^c\right],$$

where A^c denotes the complement of A, and since each term in the sum is less than $2/N^2$, then the probability of the event in (3.2.7) is bounded from below by $1/N$. It follows that to obtain a better lower bound for the probability of the event in (3.2.7) using the present techniques, a different bound for the probabilities of the event $\left\{ \left\|\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})\right\|^2 \geq (1+\epsilon)\left\|\mathbf{u}-\mathbf{v}\right\|^2 \right\}$ and $\left\{ \left\|\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})\right\|^2 < (1-\epsilon)\left\|\mathbf{u}-\mathbf{v}\right\|^2 \right\}$ must be selected. Thus, Achlioptas [3] introduces a parameter $\beta > 0$ so that for each pair $\mathbf{u}, \mathbf{v} \in V$,

$$P\left[(1-\epsilon)\left(\left\|\mathbf{u}-\mathbf{v}\right\|^2\right) \leq \left\|\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})\right\|^2 \leq (1+\epsilon)\left(\left\|\mathbf{u}-\mathbf{v}\right\|^2\right)\right] \geq 1-2/N^{2+\beta}.$$

With this choice, the probability of the event in (3.2.7) is then seen to be bounded from below by $1-1/N^{\beta}$. The parameter β becomes a fine-tuning parameter that affects the probability of the event in (3.2.7). Taking the $\beta > 0$ into account, the new expression for the Dasgupta and Gupta bound is

$$k \geq \frac{(24+12\beta)\ln N}{3\epsilon^2-2\epsilon^3}. \tag{3.2.8}$$

We will incorporate the parameter β in all the bounds presented in this work.

Next, we provide improvements to the Dasgupta and Gupta lower bound for k by 1) using the moment generating function (mgf) technique, and 2) working directly with the distribution of the random Euclidean distances.

3.3 Improvements on the Bound Provided by the Dasgupta and Gupta version of the JL Lemma.

It is possible to improve on the bound obtained from Dasgupta and Gupta's version of the JL Lemma (Eq. (3.2.8)). Two improvements are discussed in this section. Using the moment generating function (mgf) technique, two approaches provide a modest improvement on the Dasgupta and Gupta bound. However, by working directly with the exact probability distribution of the random Euclidean distances rather than the mgf, we provide a significant improvement on the Dasgupta and Gupta bound.

3.3.1 Improvement of the Dasgupta and Gupta bound using Moment Generating Function (mgf) Techniques

JL mgf Approach 1

In the proof of Dasgupta and Gupta's version of the JL Lemma, the right-tail probability is bounded above by $(g(s))^k$, where $g(s) = e^{-s(1+\epsilon)}(1-2s)^{-1/2}$, $s \in (0, 1/2)$ (Eq. (3.2.4)). The same $(g(s))^k$ is also used as an upper bound for the left-tail probability, and hence, the lower bound for k can be obtained by setting the minimized $(g(s^*))^k$ less than or equal to $1/N^{2+\beta}$. Note that the left-tail probability is bounded above by $(h(s))^k$, where $h(s)$ is strictly convex and takes the form $h(s) = e^{s(1-\epsilon)}(1+2s)^{-1/2} \leq g(s)$. Thus, one approach to improve on the lower bound for k is to find the minimized $h(s_h^*)$ separately from $g(s^*)$, where s_h^* denotes the unique minimizer of $h(s)$, and then set $(g(s^*))^k + (h(s_h^*))^k \leq 2/N^{2+\beta}$ to obtain the lower bound for k numerically.

It turns out that the minimizer s_h^* of $h(s)$ is $\frac{1}{2}\left(\frac{\epsilon}{1-\epsilon}\right) \in \left(0, \frac{1}{2}\right)$, and thus, $h(s_h^*) = exp\left(\frac{1}{2}(\epsilon + \ln(1-\epsilon))\right)$. Setting $(g(s^*))^k + (h(s_h^*))^k \leq 2/N^{2+\beta}$ yields

$$exp\left(-\frac{k}{2}(\epsilon - \ln(1+\epsilon))\right) + exp\left(\frac{k}{2}(\epsilon + \ln(1-\epsilon))\right) \leq \frac{2}{N^{2+\beta}}\,. \tag{3.3.1}$$

Thus, the lower bound for k can be obtained from Eq. (3.3.1) numerically.

JL mgf Approach 2

We examine jointly the sum of the tail probabilities instead of considering the tail probabilities separately as in the proof of Dasgupta and Gupta's version of the JL Lemma. Let $B(s) = (g(s))^k + (h(s))^k$ be the sum of the intermediate bounds for the left- and right-tail probabilities, then $B(s)$ is strictly convex since the sum of strictly convex functions is strictly convex. Denote by s_B^* the unique minimizer of $B(s)$, then $s_B^*(k)$ is the value of s that satisfies

$$e^{2sk}\left(\frac{1-2s}{1+2s}\right)^{1+k/2}\left(\frac{2s(1-\epsilon)+\epsilon}{2s(1+\epsilon)-\epsilon}\right) = 1 \ .$$

Note that $s_B^*(k)$ is a function of k. The lower bound for k is obtained numerically by finding the smallest integer k such that

$$B(s_B^*(k)) \leq 2/N^{2+\beta} \ . \tag{3.3.2}$$

Table 3.1 compares the lower bound for k for the L_2-L_2 distance using the moment generating function (mgf) technique: JL mgf Approach 1 (Eq. (3.3.1)), JL mgf Approach 2 (Eq. (3.3.2)), and Dasgupta and Gupta (DG) version of the JL Lemma. Approach 1 provides a larger improvement on the JL bound than Approach 2, but the improvement is rather modest.

The Dasgupta and Gupta (DG) bound can be improved further by working directly with the exact probability distribution of the random Euclidean distances, as provided in the next subsection.

Table 3.1 : Comparison of the lower bounds for k for L_2-L_2 distance using moment generating function (mgf) technique: JL mgf Approach 1 (Eq. (3.3.1)), JL mgf Approach 2 (Eq. (3.3.2)), and Dasgupta and Gupta (DG) version of the JL Lemma.

N(0,1) entries		Eq. (3.3.1)	Eq. (3.3.2)	**DG Bound**
N=50	$\epsilon = .1, \beta = 1$	4788	4956	5030
	$\epsilon = .3, \beta = 1$	588	592	653
	$\epsilon = .1, \beta = 2$	6425	6547	6707
	$\epsilon = .3, \beta = 2$	795	797	870
N=100	$\epsilon = .1, \beta = 1$	5656	5798	5921
	$\epsilon = .3, \beta = 1$	698	701	768
	$\epsilon = .1, \beta = 2$	7593	7689	7895
	$\epsilon = .3, \beta = 2$	943	944	1024
N=500	$\epsilon = .1, \beta = 1$	7687	7782	7991
	$\epsilon = .3, \beta = 1$	954	955	1036
	$\epsilon = .1, \beta = 2$	10319	10371	10654
	$\epsilon = .3, \beta = 2$	1285	1285	1382
N=1000	$\epsilon = .1, \beta = 1$	8566	8644	8882
	$\epsilon = .3, \beta = 1$	1065	1066	1152
	$\epsilon = .1, \beta = 2$	11497	11536	11842
	$\epsilon = .3, \beta = 2$	1432	1432	1536

3.3.2 Improvement of the Dasgupta and Gupta Bound by Working Directly with the Distribution Function of the Random Euclidean Distances

In this subsection, we provide an improvement to the bound obtained by the Dasgupta and Gupta's version of the JL Lemma by working directly with the exact probability distribution of the random Euclidean distances rather than the moment generating function technique. The following Lemma is key to proving the main result of this subsection.

Lemma 3.1 *Let k be an even integer, and $0 < \epsilon < 1$. Let $\lambda_1 = k(1+\epsilon)/2$ and $d = k/2$. Then*

$$g(k,\epsilon) = e^{-\lambda_1}\frac{\lambda_1^{d-1}}{(d-1)!}$$

is a decreasing function in k.

Proof of Lemma 3.1: Proving that $g(k+2,\epsilon) \leq g(k,\epsilon)$ is equivalent to proving that

$$(1+\epsilon)e^{-(1+\epsilon)}\left(1+\frac{1}{d}\right)^d \leq 1\ .$$

Observe that $\left(1+\frac{1}{d}\right)^d \leq e$, and thus,

$$(1+\epsilon)e^{-(1+\epsilon)}\left(1+\frac{1}{d}\right)^d \leq (1+\epsilon)e^{-\epsilon} \leq 1\ .$$

The lower bound for k can then be obtained from the following Theorem.

Theorem 3.2 *For any $0 < \epsilon < 1$, $\beta > 0$ and integer N, let k be the smallest even integer satisfying $\left(\frac{1+\epsilon}{\epsilon}\right) g(k,\epsilon) \leq \frac{1}{N^{2+\beta}}$. Then, for any set V of N points in $\mathbf{R}^p$, there is a linear map $f\colon \mathbf{R}^p \to \mathbf{R}^k$ such that for all $\mathbf{u},\mathbf{v} \in V$,*

$$P\left[(1-\epsilon)\left|\left|\mathbf{u}-\mathbf{v}\right|\right|^2 \leq \left|\left|\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})\right|\right|^2 \leq (1+\epsilon)\left|\left|\mathbf{u}-\mathbf{v}\right|\right|^2\right] \geq 1-\frac{2}{N^{2+\beta}}\ .$$

The lower bound for k can be obtained numerically by finding the smallest even integer k satisfying the inequality $\left(\frac{1+\epsilon}{\epsilon}\right) g(k,\epsilon) \leq \frac{1}{N^{2+\beta}}$.

Next, we provide the proof of Theorem 3.2.

Proof of Theorem 3.2: Recall the well-known Gamma-Poisson Relationship:

Suppose $X \sim Gamma(d,1)$, *and* $Y \sim Poisson(x)$. *Then we have* $P(X \geq x) = P(Y \leq d-1)$. *That is,*

$$\int_x^\infty \frac{1}{\Gamma(d)} z^{d-1} e^{-z} dz = \sum_{y=0}^{d-1} \frac{x^y e^{-x}}{y!} \tag{3.3.3}$$

for $d = 1, 2, 3, \ldots$.

Since $||\mathbf{y}||^2 = \sum_{j=1}^k y_j^2 \sim \chi_k^2 \stackrel{D}{=} Gamma(k/2, 2)$, using Eq. (3.3.3) with $\alpha_1 = k(1+\epsilon)$, and setting $d = k/2$, the right-tail probability can be written as,

$$P[||\mathbf{y}||^2 \geq \alpha_1] = e^{-\alpha_1/2} \sum_{y=0}^{d-1} \frac{(\alpha_1/2)^y}{y!} ,$$

and with $\alpha_2 = k(1-\epsilon)$, the left-tail probability can be written as,

$$P(||\mathbf{y}||^2 \leq \alpha_2) = e^{-\alpha_2/2} \sum_{y=d}^{\infty} \frac{(\alpha_2/2)^y}{y!} .$$

We introduce the following Theorem, which is essential in establishing the bound for the tail probabilities.

Theorem 3.3 *Let* d *be a positive integer.*

a) Let $1 \leq d < \lambda_1$. *Then,*

$$\sum_{y=0}^{d-1} \frac{\lambda_1^y}{y!} \leq \left(\frac{\lambda_1}{\lambda_1 - d}\right) \left(\frac{\lambda_1^{d-1}}{(d-1)!}\right) . \tag{3.3.4}$$

b) Let $0 < \lambda_2 < d$. *Then,*

$$\sum_{y=d}^{\infty} \frac{\lambda_2^y}{y!} \leq \left(\frac{\lambda_2}{d - \lambda_2}\right) \left(\frac{\lambda_2^{d-1}}{(d-1)!}\right) . \tag{3.3.5}$$

Proof of Theorem 3.3 Part a: Suppose $1 < d < \lambda_1$. Dividing both sides of (3.3.4) by $\left(\frac{\lambda_1}{\lambda_1 - d}\right)\left(\frac{\lambda_1^{d-1}}{(d-1)!}\right)$, it is seen that Eq. (3.3.4) is equivalent to

$$\frac{\lambda_1 - d}{\lambda_1}\left(1 + \frac{d-1}{\lambda_1} + \frac{(d-1)(d-2)}{\lambda_1^2} + \cdots + \frac{(d-1)!}{\lambda_1^{d-1}}\right) \leq 1 \,. \tag{3.3.6}$$

But

$$\begin{aligned} 1 + \frac{d-1}{\lambda_1} + \frac{(d-1)(d-2)}{\lambda_1^2} + \cdots + \frac{(d-1)!}{\lambda_1^{d-1}} &\leq \sum_{i=0}^{d-1}\left(\frac{d-1}{\lambda_1}\right)^i \\ &\leq \sum_{i=0}^{d-1}\left(\frac{d}{\lambda_1}\right)^i \\ &= \frac{1-\left(\frac{d}{\lambda_1}\right)^d}{1-\frac{d}{\lambda_1}} \end{aligned} \tag{3.3.7}$$

where (3.3.7) is obtained from the finite geometric sum.

The inequality in (3.3.6) follows immediately from (3.3.7).

Proof of Theorem 3.3 Part b: Suppose $0 < \lambda_2 < d$. Dividing both sides of (3.3.5) by $\left(\frac{\lambda_2}{d-\lambda_2}\right)\left(\frac{\lambda_2^{d-1}}{(d-1)!}\right)$, the Eq. (3.3.5) is seen to be equivalent to

$$\frac{d-\lambda_2}{\lambda_2}\frac{\lambda_2}{d}\left(1 + \frac{\lambda_2}{d+1} + \frac{\lambda_2^2}{(d+1)(d+2)} + \ldots\right) \leq 1 \,. \tag{3.3.8}$$

But,

$$\begin{aligned} 1 + \frac{\lambda_2}{d+1} + \frac{\lambda_2^2}{(d+1)(d+2)} + \cdots &\leq \sum_{i=0}^{\infty}\left(\frac{\lambda_2}{d+1}\right)^i \\ &\leq \sum_{i=0}^{\infty}\left(\frac{\lambda_2}{d}\right)^i \\ &= \frac{d}{d-\lambda_2} \,. \end{aligned} \tag{3.3.9}$$

Thus, (3.3.8) follows immediately from (3.3.9).

Using Theorem 3.3, with $\lambda_1 = \alpha_1/2 = k(1+\epsilon)/2$ and $d = k/2$, the right-tail probability is bounded as follows

$$P[||\mathbf{y}||^2 \geq \alpha_1] = e^{-\lambda_1}\sum_{y=0}^{d-1}\frac{\lambda_1^y}{y!} \leq \left(\frac{1+\epsilon}{\epsilon}\right)\left(\frac{\lambda_1^{d-1}}{(d-1)!}\right)e^{-\lambda_1} \,.$$

For the left-tail probability, setting $\lambda_2 = \alpha_2/2 = k(1-\epsilon)/2$, it follows from Theorem 3.3 that

$$\begin{aligned} P[||\mathbf{y}||^2 \leq \alpha_2] &= e^{-\lambda_2} \sum_{y=d}^{\infty} \frac{\lambda_2^y}{y!} \\ &\leq \left(\frac{1-\epsilon}{\epsilon}\right)\left(\frac{\lambda_2^{d-1}}{(d-1)!}\right) e^{-\lambda_2} \\ &\leq \left(\frac{1+\epsilon}{\epsilon}\right)\left(\frac{\lambda_1^{d-1}}{(d-1)!}\right) e^{-\lambda_1}, \end{aligned}$$

where the last inequality follows since $e^{\lambda_1-\lambda_2} \leq \left(\frac{\lambda_1}{\lambda_2}\right)^d$. Note that the bound for the left-tail probability is the same as that for the right-tail probability. Thus,

$$P[||\mathbf{y}||^2 \geq \alpha_1] + P[||\mathbf{y}||^2 \leq \alpha_2] \leq 2\left(\frac{1+\epsilon}{\epsilon}\right) g(k,\epsilon)$$

For a given ϵ, we can obtain the lower bound for k by numerically obtaining the smallest even integer k such that $\left(\frac{1+\epsilon}{\epsilon}\right) g(k,\epsilon)$ is less than or equal to $1/N^{2+\beta}$.

A numerical comparison of the bounds obtained from JL Lemma and Theorem 3.2 is presented in Table 3.2. The exact solution method numerically finds the smallest integer k such that the sum of the tail probabilities, i.e. $P[||\mathbf{y}||^2 \geq \alpha_1] + P[||\mathbf{y}||^2 \leq \alpha_2]$, is less than or equal to $2/N^{2+\beta}$. Note that the function $2\left(\frac{1+\epsilon}{\epsilon}\right) g(k,\epsilon)$ in Theorem 3.2 provides an upper bound for the sum of the tail probabilities. From Table 3.2, we observe that the lower bound for k using our approach (Theorem 3.2) is very close to the lower bound for k using the exact solution method, and significantly improves on the lower bound for k given by Dasgupta and Gupta's version of the JL Lemma. The advantage provided by our approach is reflected in the additional percentage of dimension reduction of at least 11% in all cases considered. In some of the cases, we achieve a 34% additional reduction in dimension when compared to the Dasgupta

and Gupta bound. Note that as N increases, the percentage of additional reduction provided by our approach on the Dasgupta and Gupta bound is reduced.

We next discuss a version of the JL Lemma that uses a random matrix consisting of independent and identically distributed (i.i.d.) entries of Achlioptas-type as opposed to standard Gaussians. Achlioptas [3], using Achlioptas-typed random matrix, obtained the same lower bound for k as in the case of the Gaussian random matrix, while gaining a computational speedup in the time compared to the case of the Gaussian random matrix. Section 3.5 provides improvements on the Achlioptas bound for the L_2-L_2 projection.

3.4 JL Lemma for L_2−norm with Achlioptas-typed Random Matrices

Achlioptas [3] proposed the following theorem for the lower bound for k using a random matrix consisting of i.i.d. entries drawn from the distribution provided in Eq. (3.4.1).

Achlioptas Theorem. *Let V be an arbitrary set of N points in $\mathbf{R}^p$, represented as an N x p matrix X. Given $\epsilon > 0$, $\beta > 0$, , let k be an integer satisfying*

$$k \geq \frac{(24 + 12\beta) \ln N}{3\epsilon^2 - 2\epsilon^3},$$

and let Γ be a p x k random matrix with i.i.d entries r_{ij} from the following probability distribution:

$$r_{ij} = \sqrt{q} \begin{cases} +1 & \text{with prob. } 1/(2q) \\ 0 & \text{with prob. } 1\text{-}1/q \\ -1 & \text{with prob. } 1/(2q) \end{cases} \tag{3.4.1}$$

Table 3.2 : Comparison of the lower bounds for k for L_2-L_2 distance: exact solution (numerically solving for k after setting the sum of left and right-tail probabilities equal to $2/N^{2+\beta}$), Theorem 3.2, and Dasgupta and Gupta version of the JL Lemma.

N(0,1) entries		**exact solution**	**Theorem 3.2**	**DG Bound**
N=10	$\epsilon = .1, \beta = 1$	1919	2058	2961
	$\epsilon = .3, \beta = 1$	222	254	384
	$\epsilon = .1, \beta = 2$	2792	2962	3948
	$\epsilon = .3, \beta = 2$	331	368	512
N=50	$\epsilon = .1, \beta = 1$	3776	3976	5030
	$\epsilon = .3, \beta = 1$	456	494	653
	$\epsilon = .1, \beta = 2$	5336	5572	6707
	$\epsilon = .3, \beta = 2$	654	692	870
N=100	$\epsilon = .1, \beta = 1$	4601	4822	5921
	$\epsilon = .3, \beta = 1$	561	598	768
	$\epsilon = .1, \beta = 2$	6461	6716	7895
	$\epsilon = .3, \beta = 2$	797	834	1024
N=500	$\epsilon = .1, \beta = 1$	6552	6808	7991
	$\epsilon = .3, \beta = 1$	808	846	1036
	$\epsilon = .1, \beta = 2$	9110	9390	10654
	$\epsilon = .3, \beta = 2$	1130	1168	1382
N=1000	$\epsilon = .1, \beta = 1$	7403	7670	8882
	$\epsilon = .3, \beta = 1$	916	954	1152
	$\epsilon = .1, \beta = 2$	10262	10548	11842
	$\epsilon = .3, \beta = 2$	1274	1312	1536

with $q = 1$ *or* 3. *For* $\mathbf{x} \in \mathbf{R}^p$, *define the mapping* $f\colon \mathbf{R}^p \to \mathbf{R}^k$ *by* $f(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{x}\Gamma$. *Then for all* $\mathbf{u}, \mathbf{v} \in \mathbf{R}^p$,

$$P\left[(1-\epsilon)\left|\left|\mathbf{u}-\mathbf{v}\right|\right|^2 \leq \left|\left|\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})\right|\right|^2 \leq (1+\epsilon)\left|\left|\mathbf{u}-\mathbf{v}\right|\right|^2\right] \geq 1 - \frac{2}{N^{2+\beta}}.$$

We sketch the proof of Achlioptas Theorem to facilitate the reading for subsequent sections.

Sketch of the proof of Achlioptas Theorem: Let Γ be a random matrix of dimension p x k with each entry r_{ij} from the distribution (3.4.1) with $q = 1$ or $q = 3$. Define $f(x) = \frac{1}{\sqrt{k}}\mathbf{x}\Gamma$, and $\mathbf{y} = \sqrt{k}\frac{f(\mathbf{x})}{||\mathbf{x}||}$. Then $y_j = \frac{\mathbf{x}r_j}{||\mathbf{x}||} = \sum_{i=1}^{p} c_i r_{ij}$, where $c_i = \frac{x_i}{||\mathbf{x}||} \in (-1,1)$ and $\sum_{i=1}^{p} c_i^2 = 1$, then $E(y_j) = 0$, $E(y_j^2) = 1$, and $E(||\mathbf{y}||^2) = k$.

Let $\alpha_1 = k(1+\epsilon)$, then the right-tail probability is bounded by:

$$\begin{aligned} P\left[||\mathbf{f}(\mathbf{x})||^2 \geq (1+\epsilon)\,||\mathbf{x}||^2\right] &= P\left[||\mathbf{y}||^2 \geq \alpha_1\right] \\ &\leq E\left[e^{s||\mathbf{y}||^2} e^{-s\alpha_1}\right] \quad (\text{ Markov's Inequality, s>0}) \\ &= e^{-s\alpha_1}\left(E(e^{sy_j^2})\right)^k. \end{aligned}$$

The left-hand probability can be bounded similarly with $\alpha_2 = k(1-\epsilon)$,

$$\begin{aligned} P\left[||\mathbf{f}(\mathbf{x})||^2 \leq (1-\epsilon)\,||\mathbf{x}||^2\right] &= P\left[-||\mathbf{y}||^2 \geq -\alpha_2\right] \\ &\leq e^{s\alpha_2}\left(E(e^{-sy_j^2})\right)^k, \quad s > 0 \\ &\leq e^{s\alpha_2}\left(E\left(1 - sy_j^2 + \frac{s^2 y_j^4}{2}\right)\right)^k \qquad (3.4.2) \end{aligned}$$

where Eq. (3.4.2) is obtained from the Taylor expansion for $e^{-sy_j^2}$.

The following Lemma bounds the mgf of y_j^2 by the mgf of a χ_1^2 random variable, which is the case when $r_{ij} \overset{i.i.d.}{\sim} N(0,1)$.

Achlioptas Lemma 1: *For all* $s \in [0, p/2)$, *and all* $p \geq 1$,

$$\begin{aligned} E(e^{sy_j^2}) &\leq (1-2s)^{-1/2} \\ E(y_j^4) &\leq 3 \end{aligned}$$

Note that $(1-2s)^{-1/2}$ is the moment generating function for the χ_1^2. Using Achlioptas Lemma 1, we obtain for the right-tail probability,

$$P\left[||\mathbf{y}||^2 \geq \alpha_1\right] \leq e^{-s\alpha_1}(1-2s)^{-k/2}. \tag{3.4.3}$$

Minimizing (3.4.3) with respect to s gives $s^* = \frac{1}{2}\left(\frac{\epsilon}{1+\epsilon}\right)$. Pluggin s^* back into (3.4.3) yields

$$P\left[||\mathbf{y}||^2 \geq \alpha_1\right] \leq e^{-\frac{k}{12}\left(3\epsilon^2-2\epsilon^3\right)}.$$

Similarly, using Achlioptas Lemma 1, we obtain for the left-tail probability,

$$P\left[||\mathbf{y}||^2 \leq \alpha_2\right] \leq e^{s\alpha_2}(1-s-3s^2/2)^k.$$

Taking $s = \frac{1}{2}\left(\frac{\epsilon}{1+\epsilon}\right)$ is not optimal, but it is good enough to yield,

$$P\left[||\mathbf{y}||^2 \leq \alpha_2\right] \leq e^{-\frac{k}{12}\left(3\epsilon^2-2\epsilon^3\right)}.$$

If $k \geq \frac{(24+12\beta)\ln N}{3\epsilon^2-2\epsilon^3}$, then both the left and right-tail probabilities are bounded above by $1/N^{2+\beta}$. Hence, Achlioptas Theorem is proven.

The proof of Achlioptas Lemma 1 is obtained by bounding the moments of y_j^2 by the moments of the T^2, where $T \sim N(0,1)$. As pointed out by Achlioptas [3], Achlioptas Lemma 1 fails when $q > 3$. In other words, the largest value of q for which the mgf of y_j^2 is bounded above by the mgf of T^2 is 3.

For the Rademacher random matrix (consisting of i.i.d. entries ± 1 with probability 1/2), the Achlioptas bound can be improved by taking advantage of the properties of the Rademacher random variables. Three improvements on the Achlioptas bound are provided in the next section. We first give an alternate proof of the Achlioptas Theorem.

3.5 Improvement of the Achlioptas Bound for Rademacher Random Matrices

3.5.1 Alternate Proof of Achlioptas Theorem

We should note that Achlioptas bounds the moments of y_j^2 by the moments of $T^2 \sim \chi_1^2$. Hence, the mgf of y_j^2 is bounded by the mgf of T^2. An alternate approach is provided for bounding the mgf of y_j^2 by the mgf of T^2 by working directly with the mgf of y_j.

For r_{ij}'s drawn from the distribution provided in Eq. (3.4.1), with $q = 1$ or 3,

$$M_{r_{ij}}(t) = 1 + \frac{1}{q}\left(\cosh\left(t\sqrt{q}\right) - 1\right).$$

Since $y_j = \sum_{i=1}^{p} c_i r_{ij}$, where $c_i = \frac{x_i}{||x||}$, and $\sum_{i=1}^{p} c_i^2 = 1$, we have

$$M_{y_j}(t) = \prod_{i=1}^{p}\left(1 + \frac{1}{q}\left(\cosh\left(c_i t\sqrt{q}\right) - 1\right)\right), \qquad t \in \mathbf{R}.$$

The following proposition is introduced to provide a bound on $M_{y_j}(t)$.

Proposition 3.1 *For $x \in \mathbf{R}$, and $a = 1, 2$ or 3, we have*

$$1 + \frac{1}{a}\left(\cosh\left(x\sqrt{a}\right) - 1\right) \leq e^{x^2/2}.$$

Proof of Proposition 3.1: Our proof will show that $g(x) = \frac{\log\left(1+\frac{1}{a}\left(\cosh\left(x\sqrt{a}\right)-1\right)\right)}{x^2/2}$ takes its maximum value of 1 at $x = 0$. By a symmetry argument, we only need to consider the case $x > 0$ and show that g is decreasing in $x > 0$.

For $a = 1$, $g(x) = \frac{\log(\cosh(x))}{x^2/2}$ is decreasing in $x > 0$.

For the cases $a = 2$ and $a = 3$, $g(x) = \frac{1+\frac{1}{a}\left(\cosh\left(x\sqrt{a}\right)-1\right)}{e^{x^2/2}}$. To prove that g is decreasing, we need $g'(x) < 0$.

$$g'(x) = \frac{1}{e^{x^2/2}}\left(\frac{\sqrt{a}}{a}\sinh(x\sqrt{a}) - x\left(1 + \frac{1}{a}(\cosh(x\sqrt{a}) - 1)\right)\right)$$

with $g'(0) = 0$. Let

$$h(x) = \frac{\sqrt{a}}{a}\sinh(x\sqrt{a}) - x\left(1 + \frac{1}{a}(\cosh(x\sqrt{a}) - 1)\right)$$

Since $x > 0$, and $h(0) = 0$, if $h'(x) < 0$, then $x = 0$ is maximum and $h(x) < 0$. But

$$h'(x) = \left(\frac{a-1}{a}\right)(\cosh(x\sqrt{a}) - 1) - x\frac{\sqrt{a}}{a}\sinh(x\sqrt{a})$$

with $h'(0) = 0$. Let

$$l(x) = (a-1)(\cosh(x\sqrt{a}) - 1) - x\sqrt{a}\sinh(x\sqrt{a})$$

then

$$l'(x) = \sqrt{a}(a-2)\sinh(x\sqrt{a}) - xa\cosh(x\sqrt{a})$$

with $l'(0) = 0$. For $a = 2$, we have $l'(x) = -2x\cosh(x\sqrt{2}) < 0$, which implies $g(x)$ is decreasing for $x > 0$.

For $a = 3$, let

$$m(x) = l'(x) = \sqrt{3}\sinh(x\sqrt{3}) - 3x\cosh(x\sqrt{3})$$

then $m'(x) = -3\sqrt{3}\sinh(x\sqrt{3}) < 0$, which implies $g(x)$ is decreasing for $x > 0$. Thus, Proposition 3.1 is proven. □

Using Proposition 3.1, we obtain for $t \in \mathbf{R}$, and let $a = q$ (where $q = 1$ or $q = 3$),

$$M_{y_j}(t) = \prod_{i=1}^{p}\left(1 + \frac{1}{q}\left(\cosh\left(c_i t\sqrt{q}\right) - 1\right)\right) \leq \prod_{i=1}^{p} e^{c_i^2 t^2/2} = e^{t^2/2} = M_T(t) \qquad (3.5.1)$$

where $T \sim N(0,1)$. The inequality in (3.5.1) implies for $t \in \mathbf{R}$,

$$M_{y_j^2}(t) \leq M_{T^2}(t) = (1-2t)^{-1/2}.$$

The right-tail probability is then bounded by,

$$\begin{aligned} P\left[||\mathbf{y}||^2 \geq k(1+\epsilon)\right] &\leq \left(e^{-t(1+\epsilon)} M_{y_j^2}(t)\right)^k \\ &\leq \left(e^{-t(1+\epsilon)} M_{T^2}(t)\right)^k \\ &= \left(e^{-t(1+\epsilon)} (1-2t)^{-1/2}\right)^k, \ t \in (0, 1/2). \end{aligned} \tag{3.5.2}$$

Similarly, the left-tail probability is bounded by

$$\begin{aligned} P\left[||\mathbf{y}||^2 \leq k(1-\epsilon)\right] &\leq \left(e^{t(1-\epsilon)} M_{y_j^2}(-t)\right)^k \\ &\leq \left(e^{t(1-\epsilon)} (1+2t)^{-1/2}\right)^k \\ &\leq \left(e^{-t(1+\epsilon)} (1-2t)^{-1/2}\right)^k, \ t \in (0, 1/2). \end{aligned}$$

Note that Eq. (3.5.2) is the upper bound as in the case when $r_{ij} \sim N(0,1)$, and thus, the lower bound for k for the Achlioptas-typed random matrix is the same as for the Gaussian random matrix.

For what follows, define the Rademacher random matrix as a random matrix consisting of i.i.d. entries r_{ij}'s, where

$$r_{ij} = \begin{cases} +1 & \text{with prob. } 1/2 \\ -1 & \text{with prob. } 1/2\ . \end{cases} \tag{3.5.3}$$

Next we provide improvements to the Achlioptas bound for the Rademacher random matrix for the $L_2 - L_2$ distance, by using: 1) Hoeffding's Inequality, 2) Berry-Esseen Theorem, and 3) Pinelis Inequality. We also provide the lower bound for k using a random matrix with entries following an asymmetric distribution.

The improvements on the Achlioptas bound are based on the facts that $r_{ij}^2 = 1$, and the products $r_{lmj} = r_{lj} r_{mj} \overset{D}{=} r_{ij}$ are independent ($l = 1, \ldots, p$, $m = l+1, \ldots, p$).

3.5.2 Improvement on the Achlioptas Bound using Hoeffding's Inequality

An improvement of the Achlioptas lower bound for k can be obtained from the following Theorem using Hoeffding's Inequality based on the moment generating function (mgf) technique.

Theorem 3.4 *For any $0 < \epsilon < 1$, $\beta > 0$, and integers $p \geq 2$ and N, let k be such that*

$$k \geq \left(\frac{(8+4\beta)\ln N}{\epsilon^2}\right)\left(\frac{p-1}{p}\right) . \tag{3.5.4}$$

Let Γ be a p x k Rademacher random matrix. For $\mathbf{x} \in \mathbf{R}^p$, define the mapping $f : \mathbf{R}^p \rightarrow \mathbf{R}^k$ by $f(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{x}\Gamma$. Then, for any set V of N points in $\mathbf{R}^p$, such that for all $\mathbf{u},\mathbf{v} \in V$,

$$P\left[(1-\epsilon)\left|\left|\mathbf{u}-\mathbf{v}\right|\right|^2 \leq \left|\left|\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})\right|\right|^2 \leq (1+\epsilon)\left|\left|\mathbf{u}-\mathbf{v}\right|\right|^2\right] \geq 1 - \frac{2}{N^{2+\beta}} .$$

Proof of Theorem 3.4: Let Γ be a random matrix of dimension p x k with i.i.d entries r_{ij} of Achlioptas type (3.5.3). For $\mathbf{x} \in \mathbf{R}^p$, define a linear mapping f: $\mathbf{R}^p \rightarrow \mathbf{R}^k$ by $f(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{x}\Gamma$. Let $\mathbf{y} = \sqrt{k}\frac{f(\mathbf{x})}{||\mathbf{x}||_2}$, and $y_j = \sum_{j=1}^{p} c_i r_{ij}$, where $c_i = \frac{x_i}{||\mathbf{x}||_2} \in (-1,1)$, with $\sum_{i=1}^{p} c_i^2 = 1$, we have

$$\begin{aligned} y_j^2 &= 1 + 2\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_l c_m r_{lj} r_{mj} \\ &\stackrel{D}{=} 1 + 2\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_{lm} r_{lmj} \end{aligned}$$

where $c_{lm} = c_l c_m$, and $r_{lmj} = r_{lj} r_{mj} \stackrel{D}{=} r_{ij}$. The $c_{lm} r_{lmj}$'s are independent but not identically distributed, with $E(c_{lm} r_{lmj}) = 0$, $\sigma^2 = V(c_{lm} r_{lmj}) = c_{lm}^2$, and $\rho = E(|c_{lm} r_{lmj}|^3) = |c_{lm}|^3$. This implies

$$\frac{\rho}{\sigma^3} = 1 .$$

Note that

$$y_j^4 = 1 + 4\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_l c_m r_{lj} r_{mj} + 4\left(\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_l c_m r_{lj} r_{mj}\right)^2 .$$

Thus,

$$E(y_j^2) = 1 ,$$

and

$$\begin{aligned} E(y_j^4) &= 1 + 4\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_l^2 c_m^2 \\ &\leq 1 + 2\sum_{l=1}^{p}\sum_{m=l+1}^{p} \frac{1}{p}\frac{1}{p} \qquad (3.5.5) \\ &= 1 + 2\left(\frac{p-1}{p}\right), \end{aligned}$$

where the inequality in (3.5.5) is obtained since the maximum of $\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_l^2 c_m^2$ is attained at $c_l = c_m = \frac{1}{p}$. Thus,

$$V(y_j^2) = 4\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_l^2 c_m^2 \leq 2\left(\frac{p-1}{p}\right) .$$

We can take advantage of Hoeffding's inequality (based on the minimized moment bounds on the tail probabilities) for the sum of bounded random variables to obtain a lower bound for k. Hoeffding's inequality for the tail bounds for the sum of independent bounded random variables is as follows:

Hoeffding's Inequality: *Let U_i's be independent and bounded random variables such that U_i falls in the interval $[a_i, b_i]$ $(i = 1, \ldots, m)$ with probability one. Let $S_m = \sum_{i=1}^{m} U_i$, then for any $t > 0$,*

$$P\left[S_m - E(S_m) \geq t\right] \leq e^{-2t^2/\sum_{i=1}^{m}(b_i - a_i)^2}$$

and

$$P\left[S_m - E(S_m) \leq -t\right] \leq e^{-2t^2/\sum_{i=1}^{m}(b_i-a_i)^2} .$$

Using Hoeffding's inequality, the right-tail probability is bounded by

$$\begin{aligned} P\left[||\mathbf{y}||^2 \geq k(1+\epsilon)\right] &= P\left[\sum_{j=1}^{k}\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_{lm}r_{lmj} \geq \frac{k\epsilon}{2}\right] \\ &\leq exp\left(-\frac{(2k^2\epsilon^2/4)}{4\sum_{j=1}^{k}\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_l^2 c_m^2}\right) \\ &= exp\left(-\frac{k\epsilon^2}{8\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_l^2 c_m^2}\right) \\ &\leq exp\left(-\frac{k\epsilon^2}{8\sum_{l=1}^{p}\sum_{m=l+1}^{p} \frac{1}{p}\frac{1}{p}}\right) \\ &= exp\left(-\frac{k\epsilon^2}{4}\left(\frac{p}{p-1}\right)\right) . \end{aligned} \tag{3.5.6}$$

Similarly, using Hoeffding's inequality, the left-tail probability is bounded by

$$\begin{aligned} P\left[||\mathbf{y}||^2 \leq k(1-\epsilon)\right] &= P\left[\sum_{j=1}^{k}\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_{lm}r_{lmj} \leq -\frac{k\epsilon}{2}\right] \\ &\leq exp\left(-\frac{(2k^2\epsilon^2/4)}{4\sum_{j=1}^{k}\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_l^2 c_m^2}\right) \\ &= exp\left(-\frac{k\epsilon^2}{8\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_l^2 c_m^2}\right) \\ &\leq exp\left(-\frac{k\epsilon^2}{8\sum_{l=1}^{p}\sum_{m=l+1}^{p} \frac{1}{p}\frac{1}{p}}\right) \\ &= exp\left(-\frac{k\epsilon^2}{4}\left(\frac{p}{p-1}\right)\right) . \end{aligned} \tag{3.5.7}$$

Note that Eq. (3.5.7) is the same as Eq. (3.5.6). Setting Eq. (3.5.6) less than or equal to $1/N^{2+\beta}$ yields the lower bound for k to be

$$k \geq \left(\frac{(8+4\beta)\ln N}{\epsilon^2}\right)\left(\frac{p-1}{p}\right) .$$

The percentage of additional reduction in dimensions (PARD) provided by Theorem 3.4 on the Achlioptas bound when p is large is $\frac{2}{3}\epsilon$ x 100%. Note that the PARD

is only a function of ϵ, and does not depend on N nor β. When $\epsilon = 0.1$, then the PARD = 6.7%, and when $\epsilon = 0.3$, then PARD = 20%.

Since Hoeffding's Inequality is based on the mgf technique, it may not provide the tightest of the bounds on the tail probabilities. We next work directly with the distribution of the random Euclidean distances rather than resorting to the mgf technique.

3.5.3 Improvement on the Achlioptas Bound using the Berry-Esseen Theorem based on Normal Approximations

We provide the following theorem to improve on the Achlioptas bound for the Rademacher random matrix using the Berry-Esseen Theorem based on normal approximations.

Theorem 3.5 *For any $0 < \epsilon < 1$, $\beta > 0$, and integers N, and $p \geq 2$, let k be the smallest integer satisfying*

$$1 - \Phi\left(\epsilon\sqrt{\frac{kp}{2(p-1)}}\right) + \frac{0.7915}{\sqrt{kp(p-1)/2}} \leq 1/N^{2+\beta}. \tag{3.5.8}$$

Let Γ be a p x k Rademacher random matrix. For $\mathbf{x} \in \mathbf{R}^p$, define the mapping $f : \mathbf{R}^p \rightarrow \mathbf{R}^k$ by $f(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{x}\Gamma$. Then, for any set V of N points in $\mathbf{R}^p$, such that for all $\mathbf{u}$,$\mathbf{v} \in V$,

$$P\left[(1-\epsilon)\left|\left|\mathbf{u}-\mathbf{v}\right|\right|^2 \leq \left|\left|\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})\right|\right|^2 \leq (1+\epsilon)\left|\left|\mathbf{u}-\mathbf{v}\right|\right|^2\right] \geq 1 - \frac{2}{N^{2+\beta}}\ .$$

Proof of Theorem 3.5: As in the proof of Theorem 3.4, the left- and right-tail probabilities can be written as:

$$P\left[||\mathbf{y}||^2 \geq k(1+\epsilon)\right] = P\left[\sum_{j=1}^{k}\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_{lm}r_{lmj} \geq \frac{k\epsilon}{2}\right]$$

and

$$P\left[||\mathbf{y}||^2 \leq k(1-\epsilon)\right] = P\left[\sum_{j=1}^{k}\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_{lm}r_{lmj} \leq -\frac{k\epsilon}{2}\right].$$

Using normal approximations for large k and p, the right-hand tail probability can be approximated by

$$\begin{aligned} P\left[||\mathbf{y}||^2 \geq k(1+\epsilon)\right] &= P\left[\sum_{j=1}^{k}\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_{lm}r_{lmj} \geq \frac{k\epsilon}{2}\right] \\ &\approx 1-\Phi\left(\frac{k(1+\epsilon)-k}{\sqrt{4k\sum_{l=1}^{p}\sum_{m=l+1}^{p}c_l^2c_m^2}}\right) \\ &\leq 1-\Phi\left(\frac{k(1+\epsilon)-k}{\sqrt{2k\left(\frac{p-1}{p}\right)}}\right) \\ &= 1-\Phi\left(\epsilon\sqrt{\frac{kp}{2(p-1)}}\right) \end{aligned} \tag{3.5.9}$$

Similarly, the left-tail probability can be approximated by:

$$\begin{aligned} P\left[||\mathbf{y}||^2 \leq k(1-\epsilon)\right] &\approx \Phi\left(\frac{k(1-\epsilon)-k}{\sqrt{4k\sum_{l=1}^{p}\sum_{m=l+1}^{p}c_l^2c_m^2}}\right) \\ &\leq \Phi\left(\frac{k(1-\epsilon)-k}{\sqrt{2k\left(\frac{p-1}{p}\right)}}\right) \\ &= \Phi\left(-\epsilon\sqrt{\frac{kp}{2(p-1)}}\right) \\ &= 1-\Phi\left(\epsilon\sqrt{\frac{kp}{2(p-1)}}\right) \quad . \end{aligned} \tag{3.5.10}$$

Note that Eq. (3.5.10) is the same as Eq. (3.5.9). Setting (3.5.9) $\leq 1/N^{2+\beta}$ yields the lower bound for k,

$$k \geq \left(\frac{2}{\epsilon^2}\right)\left(\frac{p-1}{p}\right)\left(\Phi^{-1}\left(1-\frac{1}{N^{2+\beta}}\right)\right)^2 . \tag{3.5.11}$$

Denote by k^* the smallest integer k satisfying Eq. (3.5.11). The k^* is obtained by using normal approximations. However, we are interested in finding an upper bound

for the tail probabilities instead of an approximation. This can be established by using Berry-Esseen Theorem, which provides an upper bound for the absolute error between the cumulative distribution function (cdf) of the sample mean and the cdf of the standard Gaussian random variable.

Berry-Esseen Theorem: *Let $X_1, \ldots, X_m$ be i.i.d. random variables with $E(X_i) = 0$, $E(X_i^2) = \sigma^2 > 0$, and $E|X_i|^3 = \rho < \infty$, for $i = 1, \ldots, m$. Also, let $\bar{X}_m$ be the normalized sample mean, and F_m the cdf of $\bar{X}_m\sqrt{m}/\sigma$, and Φ the cdf of the standard normal distribution. Then for all x and m, there exists a positive constant C such that*

$$|F_m(x) - \Phi(x)| \leq \frac{C\rho}{\sigma^3\sqrt{m}}.$$

In the case of independent random variables not necessarily identically distributed, the best C is 0.7915 (Siganov [91]), with $\sigma_i^2 = E(c_{lm}r_{lmj})^2 = c_{lm}^2$ and $\rho_i = E|c_{lm}r_{lmj}|^3 = |c_{lm}|^3$, we have $\rho_i/\sigma_i^3 = 1$. Thus, the Berry-Esseen (BE) error bound is

$$e_{BE} = \frac{0.7915}{\sqrt{kp(p-1)/2}}$$

Adjusting for the BE error bound, the lower bound for k is obtained from the following inequality,

$$1 - \Phi\left(\epsilon\sqrt{\frac{kp}{2(p-1)}}\right) + \frac{0.7915}{\sqrt{kp(p-1)/2}} \leq 1/N^{2+\beta}.$$

An additional reduction in the dimension when compared the bound obtained from Theorem 3.5 (based on the Berry-Esseen Theorem) compared to Achlioptas bound is from 10% to 40% (Table 3.5.4) for the various considered values of N, ϵ, β and p. Also, the bound obtained from Theorem 3.5 provides a larger PARD on the Achlioptas bound than the bound obtained from Theorem 3.4.

3.5.4 Improvement of the Achlioptas Bound using the Pinelis Inequality

We provide the following theorem that improves on the Achlioptas bound for the normalized sum of Rademacher random variables (Eq. (3.5.3)). For what follows, denote by Φ^{-1} the quantile function of the standard Gaussian random variable.

Theorem 3.6 *For any $0 < \epsilon < 1$, $\beta > 0$, and integers $p \geq 2$ and $N \geq 3$, let k be such that*

$$k \geq \frac{2(p-1)a_N^2}{p\epsilon^2} \ , \tag{3.5.12}$$

where $a_N = \frac{Q_N + \sqrt{Q_N^2 + 4(1.495)}}{2}$, and $Q_N = \Phi^{-1}\left(1 - \frac{1}{N^{2+\beta}}\right)$. Let Γ be a p x k Rademacher random matrix. For $\mathbf{x} \in \mathbf{R}^p$, define the mapping $f : \mathbf{R}^p \rightarrow \mathbf{R}^k$ by $f(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{x}\Gamma$. Then, for any set V of N points in $\mathbf{R}^p$, such that for all $\mathbf{u},\mathbf{v} \in V$,

$$P\left[(1-\epsilon)\left|\left|\mathbf{u}-\mathbf{v}\right|\right|^2 \leq \left|\left|\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})\right|\right|^2 \leq (1+\epsilon)\left|\left|\mathbf{u}-\mathbf{v}\right|\right|^2\right] \geq 1 - \frac{2}{N^{2+\beta}} \ .$$

The proof of Theorem 3.6 uses Pinelis Inequality [87] for tail probabilities of normalized sums of Rademacher random variables.

Proof of Theorem 3.6: As in the proof of Theorem 3.4, the left- and right-tail probabilities can be written as:

$$P\left[\left|\left|\mathbf{y}\right|\right|^2 \geq k(1+\epsilon)\right] = P\left[\sum_{j=1}^{k}\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_{lm}r_{lmj} \geq \frac{k\epsilon}{2}\right]$$

and

$$P\left[\left|\left|\mathbf{y}\right|\right|^2 \leq k(1-\epsilon)\right] = P\left[\sum_{j=1}^{k}\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_{lm}r_{lmj} \leq -\frac{k\epsilon}{2}\right]$$

We state the Pinelis Inequality [87] which aids us in establishing the bounds for the tail probabilities.

Pinelis Inequality: *Let U_i's be independent Rademacher random variables. Let $d_1, \ldots, d_m$ be any real numbers such that $\sum_i^m d_i^2 = 1$. Let $S_m = \sum_i^m d_i U_i$. Then for any $t > 0$,*

$$P\left[|S_m| \geq t\right] \leq \min\left(\frac{1}{t^2}, 2\left(1 - \Phi(t - 1.495/t)\right)\right).$$

Let $D = \sum_{j=1}^{k}\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_{lm}^2 = k\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_{lm}^2 \leq \frac{k(p-1)}{p}$, and $\tilde{c}_{lm} = \frac{c_{lm}}{\sqrt{D}}$ with $\sum_{j=1}^{k}\sum_{l=1}^{p}\sum_{m=l+1}^{p} \tilde{c}_{lm}^2 = 1$. Using Pinelis inequality, we obtain for the sum of the left- and right-tail probabilities,

$$\begin{aligned} P\left[\left|\sum_{j=1}^{k}\sum_{l=1}^{p}\sum_{m=l+1}^{p} c_{lm} r_{lmj}\right| \geq \frac{k\epsilon}{2}\right] &= P\left[\left|\sum_{j=1}^{k}\sum_{l=1}^{p}\sum_{m=l+1}^{p} \tilde{c}_{lm} r_{lmj}\right| \geq \frac{k\epsilon}{2\sqrt{D}}\right] \\ &\leq P\left[\left|\sum_{j=1}^{k}\sum_{l=1}^{p}\sum_{m=l+1}^{p} \tilde{c}_{lm} r_{lmj}\right| \geq \epsilon\sqrt{\frac{kp}{2(p-1)}}\right] \\ &\leq \min\left(\frac{1}{t^2}, 2\left(1 - \Phi(t - 1.495/t)\right)\right) \qquad (3.5.13) \end{aligned}$$

where $t = \epsilon\sqrt{\frac{kp}{2(p-1)}} > 0$. For $t \gtrapprox 1.8653$, we have $2\left(1 - \Phi(t - 1.495/t)\right) < \frac{1}{t^2}$.

Suppose we let $\frac{1}{t^2}$ to be the minimum of Eq. (3.5.13), and set it less than or equal to $2/N^{2+\beta}$, then

$$t \geq \sqrt{\frac{N^{2+\beta}}{2}}$$

For $N \geq 3$, then $\sqrt{\frac{N^{2+\beta}}{2}} > 1.8653$ for $\beta > 0$. Now, suppose $2\left(1 - \Phi(t - 1.495/t)\right)$ is the minimum of Eq. (3.5.13), and set it less than or equal to $2/N^{2+\beta}$, then

$$t \geq \frac{Q_N + \sqrt{Q_N^2 + 4(1.495)}}{2},$$

which is greater than 1.8653 for $N \geq 3$ and $\beta > 0$, with $Q_N = \Phi^{-1}\left(1 - \frac{1}{N^{2+\beta}}\right)$. Thus, setting Eq. (3.5.13) less than or equal to $2/N^{2+\beta}$ yields

$$k \geq \frac{2(p-1)a_N^2}{p\epsilon^2}$$

where $a_N = \frac{Q_N + \sqrt{Q_N^2 + 4(1.495)}}{2}$.

Table 3.5.4 compares the lower bound for k for various methods for the Rademacher random matrix using the L_2-L_2 distance: 1) Method 1 (using Hoeffding's inequality based on moment generating function technique Eq. (3.5.4)), 2) Method 2 (using Berry Esseen Theorem Eq. (3.5.8)), 3) Method 3 (using Pinelis inequality Eq. (3.5.12)), and 4) Achlioptas bound (which does not depend on p). The improvement on the Achlioptas bound provided by Method 1 is not substantial since Hoeffding's inequality is based on the mgf technique, which may not necessarily provides the tightest of the bounds. Method 3 provides a moderate percentage of additional reduction in dimension (PARD) of 15% on the Achlioptas bound when $\epsilon = 0.1$. Method 2 provides the most PARD of 10%-40%.

3.5.5 Asymmetric Simple random matrix

The Achlioptas random matrix consists of entries that have a symmetric distribution. In this subsection, we explore a random matrix consisting of entries of asymmetric distribution. In particular, the lower bound for k for the asymmetric simple random matrix is compared to the symmetric case (Achlioptas random matrix).

Suppose for $a > 0$, consider

$$r_{ij} = \begin{cases} a & \text{with prob. } \alpha = \frac{1}{1+a^2} \\ -\frac{1}{a} & \text{with prob. } 1-\alpha = \frac{a^2}{1+a^2}. \end{cases} \tag{3.5.14}$$

Note that $E(r_{ij}) = 0$, $E(r_{ij}^2) = 1$, and for $s \in \mathbf{R}$,

$$E\left(e^{sr_{ij}}\right) = \frac{1}{1+a^2}\left(e^{sa} + a^2 e^{-\frac{s}{a}}\right) .$$

Table 3.3 : Comparison of lower bound for k using Rademacher random matrix for L_2 norm with $\epsilon = 0.1$: 1) Method 1 (using Hoeffding's inequality based on moment generating function technique Eq. (3.5.4)), 2) Method 2 (using Berry Esseen Theorem Eq. (3.5.8)), 3) Method 3 (using Pinelis inequality Eq. (3.5.12)), and 4) Achlioptas bound (which does not depend on p).

	β	p	Eq. (3.5.4)	Eq. (3.5.8)	(3.5.12)	Achlioptas
$N = 10$	0.5	5000	2303	1492	2045	
		10000	2303	1492	2046	2468
$N = 10$	1	5000	2763	1912	2472	
		10000	2763	1911	2472	2961
$N = 50$	0.5	5000	3912	3009	3553	
		10000	3912	2995	3554	4192
$N = 50$	1	5000	4694	3948	4300	
		10000	4694	3821	4300	5030
$N = 100$	0.5	5000	4605	3809	4214	
		10000	4605	3715	4215	4935
$N = 100$	1	30000	5527	4816	5100	
		70000	5527	4623	5100	5921
$N = 250$	0.5	30000	5522	4807	5095	
		70000	5522	4617	5095	5916
$N = 250$	1	250000	6626	6387	6163	
		10^6	6626	5682	6163	7100

Also,

$$r_{ij}^2 = \begin{cases} a^2 & \text{with prob. } \alpha = \frac{1}{1+a^2} \\ \frac{1}{a^2} & \text{with prob. } 1-\alpha = \frac{a^2}{1+a^2} \end{cases}$$

and

$$r_{lj}r_{mj} = \begin{cases} a^2 & \text{with prob. } \alpha^2 \\ \frac{1}{a^2} & \text{with prob. } 2\alpha(1-\alpha) \\ -1 & \text{with prob. } (1-\alpha)^2. \end{cases}$$

Note that the $r_{lj}r_{mj}$'s are not independent for $a \neq 1$ ($l = 1, \ldots, p$, $m = l+1, \ldots, p$ and $j = 1, \ldots, k$) compared to the case where r_{ij}'s are of the form in Eq. (3.5.3) for $a = 1$. This is easily seen since $P(r_{1j}r_{2j} = a^2, r_{2j}r_{3j} = 1/a^2) = 0$, but $P(r_{1j}r_{2j} = a^2)P(r_{2j}r_{3j} = 1/a^2) = 2\alpha^3(1-\alpha)$. Thus, it is difficult to work directly with the distribution function of the r_{ij}'s given in Eq. (3.5.14).

The following Lemma is key to proving the ensuing Theorem which provides a lower bound for k for random matrix with i.i.d. entries drawn from the asymmetric distribution in Eq. (3.5.14).

Lemma 3.2 *For $s > 0$, let $p \geq 2$ be a positive integer, $a \in (0, 1]$, and $\epsilon \in (0, 1)$. Let*

$$A(s, a, p, \epsilon) = e^{-s\left(1+\epsilon-\frac{1}{a}\right)} \left(\cosh\left(\frac{2s}{ap}\right)\right)^{p(p-1)/2},$$

and

$$s_A^* = \frac{ap}{2} \tanh^{-1}\left(\frac{a}{p-1}\left(1+\epsilon-\frac{1}{a}\right)\right).$$

Then s_A^* is the unique minimizer of $A(s, a, p, \epsilon)$. Moreover, denote by $lA(s_A^*, a, p, \epsilon)$

the logarithm of $A(s_A^*, a, p, \epsilon)$, *then*

$$lA(s_A^*, a, p, \epsilon) = -\frac{ap}{2}\left(1+\epsilon-\frac{1}{a}\right)\tanh^{-1}\left(\frac{a}{p-1}\left(1+\epsilon-\frac{1}{a}\right)\right) + \frac{1}{2}p(p-1)\ln\left[\cosh\left(\tanh^{-1}\left(\frac{a}{p-1}\left(1+\epsilon-\frac{1}{a}\right)\right)\right)\right] .$$

Proof of Lemma 3.2: Taking ln of A,

$$lA(s, a, p, \epsilon) = -s\left(1+\epsilon-\frac{1}{a}\right) + \frac{p(p-1)}{2}\left(\ln\left(\cosh\left(\frac{2s}{ap}\right)\right)\right)$$

The derivative of $lA(s, a, p, \epsilon)$ w.r.t. s is

$$\frac{dlA(s, a, p, \epsilon)}{ds} = -\left(1+\epsilon-\frac{1}{a}\right) + \frac{p-1}{a}\tanh\left(\frac{2s}{ap}\right)$$

Setting $\frac{dlA(s,a,p,\epsilon)}{ds} = 0$ yields $s_A^* = \frac{ap}{2}\tanh^{-1}\left(\frac{a}{p-1}\left(1+\epsilon-\frac{1}{a}\right)\right)$. Taking the second derivative of $lA(.)$ gives

$$\frac{d^2lA(s, a, p, \epsilon)}{ds^2} = \frac{2(p-1)}{a^2p}\left(1-\tanh^2\left(\frac{2s}{ap}\right)\right) \geq 0$$

which implies that s_A^* is the unique minimizer of $lA(s, a, p, \epsilon)$, and hence of $A(s, a, p, \epsilon)$. Thus,

$$lA(s_A^*, a, p, \epsilon) = -\frac{ap}{2}\left(1+\epsilon-\frac{1}{a}\right)\tanh^{-1}\left(\frac{a}{p-1}\left(1+\epsilon-\frac{1}{a}\right)\right) + \frac{1}{2}p(p-1)\ln\left[\cosh\left(\tanh^{-1}\left(\frac{a}{p-1}\left(1+\epsilon-\frac{1}{a}\right)\right)\right)\right] .$$

The lower bound for k for the random matrix with i.i.d. entries drawn from the asymmetric distribution in Eq. (3.5.14) can be obtained from the following Theorem.

Theorem 3.7 *For any* $\epsilon \in (0,1)$, $\beta > 0$, $a \in \left(\frac{1}{1+\epsilon}, 1\right]$ *and integers* $p \geq 2$ *and* N, *let* k *be such that*

$$k \geq \frac{(2+\beta)\ln N}{-lA(s_A^*, a, p, \epsilon)} . \tag{3.5.15}$$

Let Γ *be a* p x k *random matrix with i.i.d. entries drawn from the asymmetric distribution in Eq.* (3.5.14). *For* $\mathbf{x} \in \mathbf{R}^p$, *define the mapping* $f : \mathbf{R}^p \rightarrow \mathbf{R}^k$ *by* $f(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{x}\Gamma$. *Then, for any set* V *of* N *points in* $\mathbf{R}^p$, *such that for all* $\mathbf{u},\mathbf{v} \in V$,

$$P\left[(1-\epsilon)\left|\left|\mathbf{u}-\mathbf{v}\right|\right|^2 \leq \left|\left|\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})\right|\right|^2 \leq (1+\epsilon)\left|\left|\mathbf{u}-\mathbf{v}\right|\right|^2\right] \geq 1 - \frac{2}{N^{2+\beta}} \ .$$

Proof of Theorem 3.7: Let Γ be a random matrix of dimension p x k with i.i.d entries r_{ij} of the form provided in Eq. (3.5.14). For $\mathbf{x} \in \mathbf{R}^p$, define a linear mapping $f\colon \mathbf{R}^p \rightarrow \mathbf{R}^k$ by $f(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{x}\Gamma$.

Define $\mathbf{y} = \sqrt{k}\frac{f(\mathbf{x})}{||\mathbf{x}||}$, and $y_j = \sum_{i=1}^{p} c_i r_{ij}$, where $c_i = \frac{x_i}{||\mathbf{x}||} \in (-1,1)$, with $\sum_{i=1}^{p} c_i^2 = 1$. Define $w_j = \sum_{i=1}^{p} c_i \tilde{r}_{ij}$, where $\tilde{r}_{ij}$ is distributed as a Rademacher random variable. Define $z_j = \frac{1}{\sqrt{p}}\sum_{i=1}^{p}\tilde{r}_{ij}$. The following Proposition provides an upper bound for the moment generating function of y_j.

Proposition 3.2 *For* $s \in \mathbf{R}$, *and* $a \in (0,1]$,

$$\begin{aligned} E\left(e^{sr_{ij}}\right) &= \frac{1}{1+a^2}\left(e^{sa} + a^2 e^{-\frac{s}{a}}\right) \\ &\leq \cosh\left(\frac{s}{a}\right) = E\left(e^{\frac{s}{a}\tilde{r}_{ij}}\right) \end{aligned} \tag{3.5.16}$$

Proof of Proposition 3.2: Proving the inequality in (3.5.16) is equivalent to proving the following:

$$g(s) = \frac{\frac{1}{1+a^2}\left(e^{sa} + a^2 e^{-\frac{s}{a}}\right)}{\cosh\left(\frac{s}{a}\right)} \leq 1 \ .$$

We show that $g(s)$ attains its maximum at $s = 0$. The derivative of $g(s)$ is

$$g'(s) \propto \left(\frac{a-1}{a}\right)e^{\frac{a+1}{a}s} + \left(\frac{a+1}{a}\right)e^{\frac{a-1}{a}s} - 2a$$

Setting $g'(s) = 0$, we obtain the maximizing $s^* = 0$. Also, it is easy to verify that $g''(s) > 0$, and hence, $s^* = 0$ is the unique maximizer of $g(s)$. Since $g(0) = 1$, Eq. (3.5.16) is proven.

Using Proposition 3.2, we have for $s \in \mathbf{R}$,

$$E(e^{sy_j}) = E\left(e^{s\sum_{i=1}^{p} c_i r_{ij}}\right) \leq E\left(e^{\frac{s}{a}\sum_{i=1}^{p} c_i \tilde{r}_{ij}}\right) = E(e^{\frac{s}{a}w_j}) \ .$$

By Achlioptas Lemma 2, we have $E(e^{sw_j}) \leq E(e^{sz_j})$. Thus, for $s \in \mathbf{R}$,

$$E\left(e^{sy_j^2}\right) \leq E\left(e^{\frac{s}{a}z_j^2}\right).$$

The right-tail probability is bounded by:

$$\begin{aligned}
P\left[||\mathbf{y}|| \geq k(1+\epsilon)\right] &\leq \left(e^{-s(1+\epsilon)}E\left(e^{sy_j^2}\right)\right)^k \ , \quad s>0 \\
&\leq \left(e^{-s(1+\epsilon)}E\left(e^{\frac{s}{a}z_j^2}\right)\right)^k \ , \quad s>0 \\
&= \left(e^{-s(1+\epsilon)}E\left(e^{\frac{s}{ap}\left(1+2\sum_{1\leq l<m\leq p}\tilde{r}_{lj}\tilde{r}_{mj}\right)}\right)\right)^k \\
&= \left(e^{-s\left(1+\epsilon-\frac{1}{a}\right)}E\left(e^{2\frac{s}{ap}\sum_{1\leq l<m\leq p}\tilde{r}_{lj}\tilde{r}_{mj}}\right)\right)^k \\
&= \left(e^{-s\left(1+\epsilon-\frac{1}{a}\right)}E\left(e^{2\frac{s}{ap}\sum_{1\leq l<m\leq p}\tilde{r}_{lmj}}\right)\right)^k \\
&= \left(e^{-s\left(1+\epsilon-\frac{1}{a}\right)}\left(\cosh\left(\frac{2s}{ap}\right)\right)^{p(p-1)/2}\right)^k
\end{aligned}$$

where $\tilde{r}_{lmj} = \tilde{r}_{lj}\tilde{r}_{mj} \stackrel{D}{=} \tilde{r}_{ij}$, and $\tilde{r}_{ij}$'s are i.i.d. Note that $\cosh(.) \geq 1$, and thus, $\frac{1}{1+\epsilon} < a < 1$ in order for $e^{-s\left(1+\epsilon-\frac{1}{a}\right)} < 1$.

Similarly, the left-tail probability is bounded by:

$$
\begin{aligned}
P\left[||\mathbf{y}|| \leq k(1-\epsilon)\right] &\leq \left(e^{s(1-\epsilon)} E\left(e^{-sy_j^2}\right)\right)^k, \quad s>0 \\
&\leq \left(e^{s(1-\epsilon)} E\left(e^{-\frac{s}{a}w_j^2}\right)\right)^k, \quad s>0 \\
&= \left(e^{s(1-\epsilon)} E\left(e^{-\frac{s}{a}\left(1+2\sum_{1\leq l<m\leq p} c_l c_m \tilde{r}_{lj}\tilde{r}_{mj}\right)}\right)\right)^k \\
&\leq \left(e^{s(1-\epsilon)} E\left(e^{-\frac{s}{ap}\left(1+2\sum_{1\leq l<m\leq p} \tilde{r}_{lj}\tilde{r}_{mj}\right)}\right)\right)^k \\
&= \left(e^{s\left(1-\epsilon-\frac{1}{a}\right)} E\left(e^{-2\frac{s}{ap}\sum_{1\leq l<m\leq p} \tilde{r}_{lj}\tilde{r}_{mj}}\right)\right)^k \\
&= \left(e^{s\left(1-\epsilon-\frac{1}{a}\right)} E\left(e^{-2\frac{s}{ap}\sum_{1\leq l<m\leq p} \tilde{r}_{lmj}}\right)\right)^k \\
&= \left(e^{s\left(1-\epsilon-\frac{1}{a}\right)} \left(\cosh\left(\frac{2s}{ap}\right)\right)^{p(p-1)/2}\right)^k \\
&\leq \left(e^{-s\left(1+\epsilon-\frac{1}{a}\right)} \left(\cosh\left(\frac{2s}{ap}\right)\right)^{p(p-1)/2}\right)^k
\end{aligned}
$$

Let $A(s,a,p,\epsilon) = e^{-s\left(1+\epsilon-\frac{1}{a}\right)}\left(\cosh\left(\frac{2s}{ap}\right)\right)^{p(p-1)/2}$, then the lower bound for k is obtained by setting the minimized $(A(s_A^*,a,p,\epsilon))^k$ less than or equal to $1/N^{2+\beta}$.

Table 3.4 compares the lower bound for k for various approaches: 1) Method 4 (asymmetric random matrix): $k \geq \frac{2\ln n}{-lA(s^*,a,p,\epsilon)}$ (Eq. (3.5.15)), 2) Method 1 (Rademacher random matrix): Theorem 3.4, and 3) Achlioptas bound (Rademacher random matrix). For method 4, the cutoff $a^* = 1/(1+\epsilon)$. As $a \to 1$, the lower bound for k using method 4 is very close to the lower bound for k using method 1. Thus, using the mgf technique, the lower bound for k obtained for the asymmetric random matrix is worse than the lower bound obtained from the symmetric random matrix (Rademacher random matrix).

Table 3.4 : Comparison of lower bound for k using Rademacher random matrix for L_2 norm with $\epsilon = 0.1$: 1) Method 4 (asymmetric random matrix): $k \geq \frac{(2+\beta)\ln N}{-lA(s^*,a,p,\epsilon)}$ (Eq. (3.5.15)), 2) Method 1: Hoeffding's bound $k \geq \frac{(8+4\beta)\ln N}{\epsilon^2}$ (Eq. (3.5.4)), and 3) Achlioptas bound. For Method 4, the cutoff $a^* = 1/(1+\epsilon) = 0.909$.

			Method 4			Method 1	Achlioptas
	β	p	$a = 0.99$	0.999	$a = 0.9999$		
$N = 50$	0.5	5000	4938	3999	3920	3912	
		10000	4939	4000	3921	3912	4192
$N = 50$	1	5000	5926	4799	4704	4694	
		10000	5926	4799	4705	4694	5030
$N = 100$	0.5	5000	5813	4708	4615	4605	
		10000	5814	4708	4615	4605	4935
$N = 100$	1	30000	6977	5650	5539	5527	
		700000	6977	5650	5539	5527	5921
$N = 250$	0.5	30000	6971	5645	5534	5522	
		70000	6971	5645	5534	5522	5916
$N = 250$	1	250000	8354	6767	6632	6626	
		10^6	8354	6767	6632	6626	7100

3.6 Extending the JL Lemma Using the L_1-Norm

The Johnson-Lindenstrauss (JL) Lemma states that a set of N points in any Euclidean space can be mapped to a Euclidean space of dimension $k = O(\ln N/\epsilon^2)$ such that the pairwise distance between the points are preserved within a factor of $1 \pm \epsilon$. Since the L_1 distance is more robust against outliers than the L_2 distance, it is of interest to explore the effect of Random Projection on dimension reduction using the L_1 norm. In other words, a linear mapping for a set of N points from $p-$dimensional space to $k = O(\ln N/\epsilon^2)$ dimensional space is desirable so that the pairwise L_1 distances between the points are preserved within a factor of $1 \pm \epsilon$. However, due to the results of Brinkman and Charikar [19], Charikar and Sahai [24], Lee and Naor [63], and Indyk [51], the JL Lemma cannot be extended to the L_1 norm using a linear mapping.

3.7 RP: L_2-L_1 Norm with the Normal Random Matrix

Although it is not possible in the case of a linear mapping to obtain a totally satisfying result when the L_1 norm is used to measure distances in both the space of points to be projected and the space of the projected points, it is possible to obtain good results by using the L_2 norm in the space of points to be projected and the L_1 norm to measure distance between the projected points, as discussed next.

In this subsection, a theorem for the linear projection of N points in $p-$dimensional space onto a $k-$dimensional space using i.i.d. standard Gaussians as entries of the random matrix Γ is presented where the L_2 norm is used as a distance in the original space, and L_1 is used as a distance in the $k-$dimensional target space. It turns out that the L_2 pairwise distances of the original points are preserved within a factor of $(1 \pm \epsilon)\sqrt{2/\pi}$ of the L_1 distances of the projected points. For the same factor of

$(1\pm\epsilon)\sqrt{2/\pi}$, Ailon and Chazelle [4] (sparse Gaussian random matrix with fast Fourier transform) and Matousek [75] (sparse Achlioptas-typed random matrix) obtain the lower bound for k to be:

$$k \geq C\epsilon^{-2}(2\ln(1/\delta))$$

where $\delta \in (0,1)$, $\epsilon \in (0,1/2)$, and C is a sufficiently large constant. Here, δ is a parameter that relates to the probability with which any two projected points remain within $(1 \pm \epsilon)\sqrt{2/\pi}$ of the L_2 distance of the original points. Although the multiplicative constant C is not provided, it was taken to be 1 in one of the proofs in Matousek [75]. When $\delta = 1/N^{2+\beta}$, then $k = O\left(\frac{(4+2\beta)\ln N}{\epsilon^2}\right)$.

The following Theorem gives an improvement on the lower bound for k provided by Ailon and Chazelle [4] and Matousek [75].

In what follows, for $s > 0$, let $A(s) = 2e^{-s\sqrt{2/\pi}(1+\epsilon)+s^2/2}\Phi(s)$. For a given $\epsilon \in (0,1)$, let $s^*(\epsilon)$ be the value that minimizes $A(s)$. Equivalently, let s^* be the unique solution to $s = \sqrt{2/\pi}(1+\epsilon) - \frac{\phi(s)}{\Phi(s)}$.

Theorem 3.8 *For any $0 < \epsilon < 1$, $\beta > 0$ and any positive integer N, let k be such that*

$$k \geq \frac{(2+\beta)\ln N}{-\ln(A(s^*))} \,. \tag{3.7.1}$$

Let Γ be a p x k random matrix with i.i.d. standard Gaussian entries. For $\mathbf{x} \in \mathbf{R}^p$, define the mapping $f : \mathbf{R}^p \to \mathbf{R}^k$ by $f(\mathbf{x}) = \frac{1}{k}\mathbf{x}\Gamma$. Then, for any set V of n points in $\mathbf{R}^p$, such that for all $\mathbf{u},\mathbf{v} \in V$,

$$P\left[(1-\epsilon)\sqrt{\frac{2}{\pi}}\left(||\mathbf{u}-\mathbf{v}||_2\right) \leq ||\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})||_1 \leq (1+\epsilon)\sqrt{\frac{2}{\pi}}\left(||\mathbf{u}-\mathbf{v}||_2\right)\right] \geq 1 - \frac{2}{N^{2+\beta}} \,.$$

Proof of Theorem 3.8: Let Γ be a random matrix of dimension p x k with i.i.d entries $r_{ij} \sim N(0,1)$. For $\mathbf{x} \in \mathbf{R}^p$, define a linear mapping $f\colon \mathbf{R}^p \to \mathbf{R}^k$ by $f(\mathbf{x}) =$

$\frac{1}{k}\mathbf{x}\Gamma$. Let

$$y_j = \frac{\mathbf{x}r_j}{||\mathbf{x}||_2} \sim N(0,1) \ .$$

Then, $E(||\mathbf{y}||_1) = k\sqrt{2/\pi}$, and $M_{|y_j|}(s) = 2e^{s^2/2}\Phi(s)$.

Let $\alpha_1 = k\sqrt{2/\pi}(1+\epsilon)$, then the right-tail probability is bounded by

$$\begin{aligned} P\left[||\mathbf{f}(\mathbf{x})||_1 \geq \sqrt{2/\pi}(1+\epsilon)\,||\mathbf{x}||_2\right] &= P\left[||\mathbf{y}||_1 \geq \alpha_1\right] \\ &\leq \left(2e^{-(s\alpha_1/k)+(s^2/2)}\Phi(s)\right)^k \ , \quad s>0. \end{aligned}$$

Let $A(s) = e^{-(s\alpha_1/k)+(s^2/2)}\Phi(s)$, and denote by s^* the minimizer of A, so that s^* is the solution to

$$s = \sqrt{2/\pi}(1+\epsilon) - \frac{\phi(s)}{\Phi(s)} \ .$$

The second derivative of $A(s)$ with respect of s is taken to ensure that s^* is the minimizer of A.

$$A''(s) = e^{-(s\alpha_1/k)+(s^2/2)}\left[\left(\left(s-\frac{\alpha_1}{k}\right)^2+1\right)\Phi(s) + \left(s-2\frac{\alpha_1}{k}\right)\phi(s)\right] \ .$$

Note that for $s>0$,

$$\left(s-\frac{\alpha_1}{k}\right)^2 + 1 > 2\left(\frac{\alpha_1}{k}\right)\left(\frac{\phi(s)}{\Phi(s)}\right) \ .$$

Thus, $A''(s) > 0$, which implies s^* is the unique minimizer of A. Setting $A(s^*) \leq 1/N^{2+\beta}$, we obtain the lower bound for k to be $k \geq \frac{(2+\beta)\ln N}{-\ln A(s^*)}$.

Similarly, let $\alpha_2 = k\sqrt{2/\pi}(1-\epsilon)$, then the left-tail probability is bounded by

$$\begin{aligned} P\left[||\mathbf{f}(\mathbf{x})||_1 \leq \sqrt{2/\pi}(1-\epsilon)\,||\mathbf{x}||_2\right] &= P[||\mathbf{y}||_1 \leq \alpha_2] \\ &\leq \left(2e^{(s\alpha_2/k)+(s^2/2)}\left(1-\Phi(s)\right)\right)^k \ , \quad s>0 \ . \end{aligned}$$

Let

$$B(s) = 2e^{(s\alpha_2/k)+(s^2/2)}\left(1-\Phi(s)\right) \ .$$

The next proposition provides $B(s) \leq A(s)$.

Proposition 3.3 *For all $\zeta > 0$, we have*

$$e^{2\sqrt{2/\pi}\zeta} < \frac{\Phi(\zeta)}{1-\Phi(\zeta)} \ . \tag{3.7.2}$$

Proof of Proposition 3.3: Let $f(\zeta) = \frac{\Phi(\zeta)}{1-\Phi(\zeta)} e^{-2\zeta\sqrt{2/\pi}}$, then Eq. (3.7.2) is equivalent to

$$f(\zeta) > 1 \tag{3.7.3}$$

It suffices to prove that $f(\zeta)$ is an increasing function. Taking the derivative of f with respect to ζ yields

$$f'(\zeta) = \frac{e^{-2\zeta\sqrt{2/\pi}}}{1-\Phi(\zeta)} \left[\frac{\phi(\zeta)}{1-\Phi(\zeta)} - 2\sqrt{2/\pi}\Phi(\zeta) \right] \ .$$

We should note that the first term is positive. The ratio $\frac{\phi(\zeta)}{1-\Phi(\zeta)}$ is the inverse of the Mill's ratio, which is an increasing function, and we observe that

$$\frac{\phi(\zeta)}{1-\Phi(\zeta)} > 2\sqrt{2/\pi}\Phi(\zeta)$$

which implies $f'(\zeta) > 0$, and hence, f is an increasing function of ζ. The minimum of f is attained when $\zeta = 0$. In other words, $\min_{\zeta} f(\zeta) = 1$, and hence Eq. (3.7.3) is proven.

Using Proposition 3.3 with $\zeta = s$, $B(s) \leq A(s)$ for $s > 0$. Thus, the left-tail probability is bounded by

$$P\left[||\mathbf{f}(\mathbf{x})||_1 \leq \sqrt{2/\pi}(1-\epsilon)\,||\mathbf{x}||_2 \right] \leq \left(2e^{-(s\alpha_1/k)+(s^2/2)}\Phi(s) \right)^k \ . \tag{3.7.4}$$

Note that the right side of inequality (3.7.4) for the left-tail probability is the same as in the case for the right-tail probability.

Table 3.5 compares the lower bound for k obtained from Ailon and Chazelle [4] and Matousek [75] for L_2-L_1 distance ($C = 1$), and Theorem 3.8 for L_2-L_1 distance.

The random matrix has i.i.d. standard Gaussian entries. We observe that the lower bounds for k from Ailon and Chazelle [4] and Matousek [75] are significantly larger than the lower bound for k obtained from Theorem 3.8. In most cases, the results of Theorem 3.8 provide an additional reduction of $36\% - 40\%$ in the lower bound for k.

Table 3.5 : Normal random matrix: comparison of the lower bounds for k from Matousek [75] for L_2-L_1 distance ($C = 1$), and Theorem 3.8 for L_2-L_1 distance.

N(0,1) entries		$L_2 - L_1$ **Matousek**	$L_2 - L_1$ **Theorem 3.8**
N=50	$\epsilon = .1, \beta = 1$	2348	1398
	$\epsilon = .3, \beta = 1$	261	168
	$\epsilon = .1, \beta = 2$	3130	1863
	$\epsilon = .1, \beta = 2$	348	223
N=100	$\epsilon = .1, \beta = 1$	2764	1645
	$\epsilon = .3, \beta = 1$	308	197
	$\epsilon = .1, \beta = 2$	3685	2193
	$\epsilon = .1, \beta = 2$	410	263
N=500	$\epsilon = .1, \beta = 1$	3729	2220
	$\epsilon = .3, \beta = 1$	415	266
	$\epsilon = .1, \beta = 2$	4972	2960
	$\epsilon = .1, \beta = 2$	553	354
N=1000	$\epsilon = .1, \beta = 1$	4145	2468
	$\epsilon = .3, \beta = 1$	461	296
	$\epsilon = .1, \beta = 2$	5527	3290
	$\epsilon = .1, \beta = 2$	615	394

3.7.1 RP: L_2-L_1 Norm with the Achlioptas-typed Random Matrix

The following Corollary provides an extension to Theorem 3.8 to the case where the entries of Γ are drawn from the Achlioptas types of distribution (eq. (3.1.3) with $q = 1$ or $q = 3$).

Corollary 3.2

For any $0 < \epsilon < 1$, $\beta > 0$, and any positive integer N, let k be as in Eq. (3.7.1) of Theorem 3.8. Let Γ be a p x k random matrix with i.i.d. entries drawn from one of Achlioptas distributions (eq. (3.1.3) with $q = 1$ or $q = 3$). For $\mathbf{x} \in \mathbf{R}^p$, define the mapping $f : \mathbf{R}^p \rightarrow \mathbf{R}^k$ by $f(\mathbf{x}) = \frac{1}{k}\mathbf{x}\Gamma$. Then, for any set V of N points in $\mathbf{R}^p$, such that for all $\mathbf{u},\mathbf{v} \in V$,

$$P\left[(1-\epsilon)\sqrt{\frac{2}{\pi}}\left(||\mathbf{u}-\mathbf{v}||_2\right) \leq ||\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v})||_1 \leq (1+\epsilon)\sqrt{\frac{2}{\pi}}\left(||\mathbf{u}-\mathbf{v}||_2\right)\right] \geq 1-\frac{2}{N^{2+\beta}} .$$

Proof of Corollary 3.2: Let Γ be a random matrix of dimension p x k with i.i.d entries from an Achlioptas distribution ($q = 1$ or $q = 3$). For $\mathbf{x} \in \mathbf{R}^p$, define a linear mapping f: $\mathbf{R}^p \rightarrow \mathbf{R}^k$ by $f(\mathbf{x}) = \frac{1}{k}\mathbf{x}\Gamma$. Let

$$y_j = \frac{\mathbf{x}r_j}{||\mathbf{x}||_2} = \sum_{i=1}^{p} c_i r_{ij} ,$$

where $c_i = \frac{x_i}{||\mathbf{x}||_2}$, so that $\sum_{i=1}^{p} c_i^2 = 1$. Then, $E(||\mathbf{y}||_1) = k\sqrt{2/\pi}$, and

$$M_{y_j}(t) = \prod_{i=1}^{p}\left(1+\frac{1}{q}\left(\cosh(c_i t\sqrt{q})-1\right)\right) , \forall t .$$

Using Proposition 3.1, for $t \in \mathbf{R}$, $q = 1$ or 3, and $Z \sim N(0,1)$,

$$M_{y_j}(t) = \prod_{i=1}^{p}\left(1+\frac{1}{s}\left(\cosh\left(c_i t\sqrt{s}\right)-1\right)\right) \leq \prod_{i=1}^{p} e^{c_i^2 t^2/2} = e^{t^2/2} = M_Z(t) . \quad (3.7.5)$$

The inequality in (3.7.5) implies

$$M_{|y_j|}(t) \leq M_{|Z|}(t) = 2e^{t^2/2}\Phi(t) , t \in \mathbf{R} .$$

Thus, for $\alpha_1 = k\sqrt{2/\pi}(1+\epsilon)$, the right-tail probability is bounded by

$$\begin{aligned} P\left[||\mathbf{f}(\mathbf{x})||_1 \geq \sqrt{2/\pi}(1+\epsilon)\,||\mathbf{x}||_2\right] &= P\left[||\mathbf{y}||_1 \geq \alpha_1\right] \\ &\leq \left(2e^{-(s\alpha_1/k)} M_{|y_j|}(s)\right)^k \;, \quad s>0 \\ &\leq \left(2e^{-(s\alpha_1/k)} M_{|Z|}(s)\right)^k \\ &\leq \left(2e^{-(s\alpha_1/k)+(s^2/2)}\Phi(s)\right)^k \qquad (3.7.6) \end{aligned}$$

where the last inequality (3.7.6) is the same as in the case of the Gaussian random matrix.

Similarly, for $\alpha_2 = k\sqrt{2/\pi}(1-\epsilon)$, the left-tail probability is bounded by

$$\begin{aligned} P\left[||\mathbf{f}(\mathbf{x})||_1 \leq \sqrt{2/\pi}(1-\epsilon)\,||\mathbf{x}||_2\right] &= P\left[||\mathbf{y}||_1 \leq \alpha_2\right] \\ &\leq \left(2e^{(s\alpha_2/k)} M_{|y_j|}(-s)\right)^k \;, \quad s>0 \\ &\leq \left(2e^{(s\alpha_2/k)} M_{|Z|}(-s)\right)^k \\ &\leq \left(2e^{(s\alpha_2/k)+(s^2/2)}\left(1-\Phi(s)\right)\right)^k \\ &\leq \left(2e^{-(s\alpha_1/k)+(s^2/2)}\Phi(s)\right)^k \;. \end{aligned}$$

The last inequality is the same as in the case of the right-tail probability, and hence, we are done.

Note that the lower bound for k using the Achlioptas-typed random matrix is the same as the lower bound for k using the Gaussian random matrix. The proof of Corollary 3.2 follows from Theorem 3.8 after bounding the moment generating function (mgf) of a Achlioptas-typed random variable by the mgf of a standard Gaussian random variable.

3.8 Discussion

Random Projection (RP) has emerged as a powerful yet computationally simple method of dimension reduction. The main motivation for RP is the Johnson-Lindenstrauss (JL) Lemma, which states that N points in high-dimensional space can be projected onto a $k = O(\ln N/\epsilon^2)$-dimensional space such that the pairwise Euclidean distances between any two points is preserved within a factor of $1 \pm \epsilon$. In this chapter, we revisit the JL Lemma when the random matrices are Gaussian or of Achlioptas type, focusing on improving the lower bound for k. For the Gaussian random matrix, we provide improvements on the lower bound for k obtained from Dasgupta and Gupta's version of the JL Lemma for the L_2-L_2 distance by 1) using the moment generating function (mgf) technique and 2) working directly with the exact distribution of the random Euclidean distances ($11 - 34\%$ additional reduction on Dasgupta and Gupta bound). For the Gaussian random matrix using the L_2-L_1 distance, we provide an improvement on the Matousek bound of $36 - 40\%$. For the Rademacher random matrix using L_2-L_2 distance, three improvements on the Achlioptas bound are provided based on 1) Hoeffding's inequality, 2) Berry-Esseen Theorem ($10 - 40\%$ additional reduction on the Achlioptas bound), and 3) Pinelis inequality. For the Achlioptas random matrix using the L_2-L_1 distance, $36 - 40\%$ additional reduction on the Matousek bound is achieved.

Despite such promising improvements, the lower bound for k is still quite large for practical purposes. However, several papers in the literature such as Bingham and Mannila [16], Fradkin and Madigan [38], and Goel et al. [41] use a value for k that is much smaller than the bound provided by Dasgupta and Gupta's version of the JL Lemma, and the results obtained from Random Projection (RP) are comparable to those from Principal Component Analysis (PCA) in terms of classification accuracy.

The advantage of RP over PCA is in the computational savings when k is moderately large, i.e. $k > 50$. Since the empirical evidence suggests that a much smaller k than the Dasgupta and Gupta bound, this area of research is still open to new findings in terms of the improvements on the lower bound for k.

In the next chapter, we propose the method of Rank-based Modified Partial Least Squares (RMPLS). Similar to Principal Component Analysis (PCA) and Partial Least Squares (PLS), RMPLS is based on an optimization criterion as opposed to Random Projection (RP) which uses the criterion of preserving pairwise distances among the points with a small distortion when projecting points from high to low-dimensional space.

Chapter 4

Rank-based Modified Partial Least Squares (RMPLS)

In this chapter, a variant of Partial Least Squares (PLS) is proposed. The Rank-based Modified Partial Least Square (RMPLS) is insensitive to outlying values of the response and covariates, and incorporates the censoring information. The derivation of the weight vectors of RMPLS as solution to an optimization criterion is provided. Also, an assessment of the performance of the different dimension reduction methods is provided based on simulation work and real datasets using the Cox and AFT models as the regression model.

Next, we set forth the notation used throughout this chapter. For a vector $z = (z_1, \ldots, z_N)$, denote by the ranks of the elements of z as the indices of the positions of the z_i's in either ascending or descending order. For a series of N measurements of $u = (u_1, \ldots, u_N)$ and $v = (v_1, \ldots, v_N)$, define the sample Pearson correlation coefficient between u and v as

$$Cor(u, v) = \frac{\sum_{i=1}^{N}(u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\sum_{i=1}^{N}(u_i - \bar{u})^2}\sqrt{\sum_{i=1}^{N}(v_i - \bar{v})^2}},$$

where $\bar{u} = \frac{1}{N}\sum_{i=1}^{N} u_i$, and $\bar{v} = \frac{1}{N}\sum_{i=1}^{N} v_i$. The sample Spearman correlation coefficient between u and v, denoted by $Cor_R(u, v)$, is the sample Pearson correlation coefficient between the ranks of u and the ranks of v. Since the Spearman correlation coefficient is based on the ranks, it is robust against outliers in both u and v.

We next discuss the method of Rank-based Modified Partial Least Squares, which

uses the Spearman rank correlation coefficient in the optimization criterion.

4.1 The Method of RMPLS

The optimization criteria of PLS involves the usual Pearson correlation coefficient between a linear combination of the covariates X and the response y. The Pearson correlation is nonrobust against outliers in either the response or covariates (Romanazzi [89]). To cope with outliers in X and y, we propose using a correlation measure based on the ranks, which is insensitive to outliers. In other words, the usual Pearson correlation is replaced by the Spearman rank correlation.

The orthogonal scores algorithm in chapter 2 is provided for the Partial Least Squares (PLS) procedure with the standardized covariate data matrix X and the response y. Since the Spearman rank correlation is the Pearson correlation on the ranks, the orthogonal scores algorithm needs to be modified to incorporate the ranks of the columns of X and the ranks of y. In step 2 of the algorithm, since $Cor(X, y) = X^T y$, we replace $Cor(X, y)$ with $Cor_R(X, y)$ where $Cor_R(X, y)$ denotes the Pearson correlation of the ranks between the columns of the matrix X and the vector y. In step 4, q_2 can be expressed as $q_2 = X^T t = \frac{X^T X w}{\tilde{t}^T \tilde{t}}$. Since $Cor(X, X) = X^T X$, we make the change $q_2 = \frac{Cor_R(X,X)w}{\tilde{t}^T \tilde{t}}$. In step 5, we update R_X and R_y instead of X and y. Here, the columns of R_X correspond to the ranks of the columns of X and R_y denotes the ranks of y. To incorporate the censoring information, we use the method of Modified Partial Least Squares (MPLS) of Nguyen and Rocke [79] with these changes under the Cox model, and Reweighted PLS (RWPLS) and Mean-Imputation PLS (MIPLS) of Datta et al. [30] with these changes under the Accelerated Failure Time (AFT) model. For MPLS, the q_2's in step 4 of the orthogonal scores algorithm are the same as the dot product a's mentioned by Nguyen and Rocke [79]. We denote the rank-based

methods of PLS as RMPLS, RRWPLS and RMIPLS, respectively.

The weight vectors w_k's in RMPLS can be derived as a solution to an optimization problem (see Nguyen and Rojo [81, 82] for details). The censoring is ignored for simplicity (the censoring is incorporated using the MPLS procedure of Nguyen and Rocke [79], and RWPLS and MIPLS procedures of Datta et al. [30]). The criterion of the usual PLS is to find the weight vector, w, such that w maximizes the covariance of Xw and y. An equivalent statement in terms of the ranks is to find the weight vector, w, such that w maximizes the covariance of R_{Xw} and R_y, where R_z denotes the ranks of the vector z. RMPLS explores a different optimization problem. The columns of the data matrix X and the response y are first converted to their ranks and then centered, denoted by R_X and R_y respectively. We search for the weight vector w such that w maximizes the covariance of $R_X w$ and R_y. The first weight vector, w_1, is obtained from the following maximization criterion,

$$w_1 = \arg\max_{w^T w=1} w^T \, Cov_R(X, y) = \arg\max_{w^T w=1} (N-1)^{-1} w^T R_X^T R_y$$

where Cov_R is the covariance of the ranks, R_X is the matrix of the ranks of X (i.e., columns of R_X correspond to the ranks of the columns of X), and R_y is the vector of the ranks of y. Here, R_X and R_y are centered.

Using Corollary A.2, with $B = I$, $x = w$, and $a = R_X^T R_y$, we obtain

$$w_1 = \frac{R_X^T R_y}{||R_X^T R_y||}$$

The first component is $\tilde{x}_1 = X w_1$. The second weight vector, w_2, is obtained from the following maximization criterion,

$$w_2 = \arg\max_{w^T w=1} w^T \, Cov_R(X, y) = \arg\max_{w^T w=1} (N-1)^{-1} w^T R_X^T R_y$$

subject to the constraint $w^T X^T \tilde{x}_1 = 0$.

Let $S_X = X^T X$, and $S_{R_X} = R_X^T R_X$. We can deduce that

$$w_2 \propto \left(I - \frac{w_1^T S_X w_1}{w_1^T S_{R_X} S_X w_1} S_{R_X} \right) w_1.$$

where I is a p x p identity matrix, and $\frac{w_1^T S_X w_1}{w_1^T S_{R_X} S_X w_1}$ is a constant. We should note that

$$\begin{aligned} w_2^T X^T \tilde{x}_1 = w_2^T S_X w_1 &= w_1^T S_X w_1 - \frac{w_1^T S_X w_1}{w_1^T S_{R_X} S_X w_1} w_1^T S_{R_X} S_X w_1 \\ &= w_1^T S_X w_1 - w_1^T S_X w_1 = 0. \end{aligned}$$

In general, the k^{th} weight vector is obtained from the following maximization criterion,

$$w_k = \underset{w^T w=1}{\arg\max}\ w^T\ Cov_R(X, y) = \underset{w^T w=1}{\arg\max}\ (N-1)^{-1} w^T R_X^T R_y$$

subject to $w_k^T S_X w_j = 0$, for $j = 1, \ldots, k-1$.

It turns out that w_k, $k \geq 2$, takes the form

$$w_k \propto P_{k-1} w_1$$

where

$$P_{k-1} = I - \zeta_1 S_{R_X} - \zeta_2 S_{R_X}^2 - \cdots - \zeta_{k-1} S_{R_X}^{k-1}$$

where $S_{R_X}^j = \underbrace{S_{R_X} S_{R_X} \ldots S_{R_X}}_{j \text{ times}}$, and $\zeta_1, \zeta_2, \ldots, \zeta_{k-1}$ can be obtained by solving the following system of linear equations for ζ's,

$$\begin{aligned} w_1^T P_{k-1} S_X w_1 &= 0 \\ w_1^T P_{k-1} S_X w_2 &= 0 \\ &\vdots \\ w_1^T P_{k-1} S_X w_{k-1} &= 0. \end{aligned}$$

An assessment of the performance of the different dimension reduction methods is provided next. The Cox Proportional Hazards (PH) model and the Accelerated Failure Time (AFT) model are discussed first. Details on the simulation procedure is provided in Appendix A.

4.2 Assessment of the Dimension Reduction Methods

The performance of several dimension reduction methods is assessed via a simulation study. In particular, the rank-based versions of modified PLS are compared to their unranked counterparts as well as several other leading dimension reduction methods. Details of the simulation procedure are provided in Appendix A.

Because of the high dimensionality of the microarray gene expression data, the usual regression techniques cannot be applied directly to the data. It is necessary to first reduce the high dimensionality of the microarray data from N x p to N x k, such that $k < N \ll p$, and then apply an appropriate regression technique to the reduced data.

Collected along with the microarray gene expression data is the survival information on the patients, which also includes the censoring. Censoring arises when an individual's lifetime is known to occur only in a certain period of time (Klein and Moeschberger [59]). For right-censored data, the individual's lifetime is only known to exceed a given time, which occurs if the individual is still alive at the end of the study or is lost to follow-up at any time during the study (Leung et al. [64]). This work focuses on right-censored data. Details regarding the different types of censoring are provided in Appendix A.

When censoring is present, the actual survival times y are not always observable. Instead, N independent triplets of (T_i, δ_i, X_i), for $i = 1, \ldots, N$, are observed, where $T_i = \min(y_i, c_i)$, $\delta_i = I(y_i \leq c_i)$ is the censoring indicator ($\delta_i = 0$ if the i^{th} individual is censored, and $\delta_i = 1$ otherwise), and c_i's are the censoring times. Conditioned on the covariates X, y and c are assumed to be independent. A popular regression model that incorporates the censoring information is the Cox proportional hazards model [26]. Another popular model is the Accelerated Failure Time (AFT) model. We now

describe these models in detail.

4.2.1 Regression Models for Censored Responses

The hazard function $h(t)$ measures the instantaneous risk of non-survival in the next small time interval, given survival up to time t. Mathematically, $h(t)$ is expressed as:

$$h(t) = \lim_{dt \to 0} \frac{P\left[t < T < t + dt | T > t\right]}{dt}$$

where T is a random variable denoting the time of death. The survival function $S(t) = P[T > t]$ is the probability of surviving beyond time t. The hazard function can be expressed in terms of the survival function,

$$h(t) = \frac{f(t)}{S(t)}$$

where $f(t)$ is the density function. Alternatively, we can express the cumulative hazard function $H(t)$, defined as $H(t) = \int_0^t h(s)ds$, in terms of survival function,

$$H(t) = -log(S(t))$$

The Cox Proportional Hazards (PH) model and the Accelerated Failure Time (AFT) model are discussed next. Both regression models require the number of covariates to be smaller than the number of cases. However, the number of genes (covariates) far exceed the number of cases for microarray gene expression data. One approach to cope with the high dimensionality of the microarray data is to apply dimension reduction methods to the original data, and then use the appropriate regression model with the reduced data matrix.

Cox Proportional Hazards (PH) Model

The Cox PH model expresses the hazard function as follows,

$$h(t, X_n; \beta) = h_0(t) e^{X_n' \beta} \tag{4.2.1}$$

where $h_0(t)$ denotes an unspecified baseline hazard function, and $X_n = (X_{n1}, \ldots, X_{np})$ are the covariates corresponding to the n^{th} individual, and $\beta = (\beta_1, \ldots, \beta_p)$ are the regression coefficients. Linearizing (4.2.1) by dividing by $h_0(t)$ and taking logarithm on both sides, we obtain

$$log\left(\frac{h(t, X_n; \beta)}{h_0(t)}\right) = X_n'\beta. \tag{4.2.2}$$

Equations (4.2.1) and (4.2.2) imply two assumptions. The first assumption is the *proportionality of the hazard rates.* In other words, given two individuals with different covariate values, the ratio of the hazard functions for these two individuals does not depend on time. The second assumption is the log-linear relationship between the covariates and the underlying hazard function. We should note that the covariates in the Cox model (4.2.1) act on a multiplicative scale.

Using the sample data, the Cox partial likelihood can written as:

$$L(\beta) = \prod_{i=1}^{D} \frac{e^{X_{(i)}'\beta}}{\sum_{j \in R(t_i)} e^{X_j'\beta}} \tag{4.2.3}$$

where D is the number of deaths, $t_1 < t_2 < \cdots < t_D$ are the ordered death times, $X_{(i)}$ are the covariates corresponding to the individual with survival time t_i, and the risk set $R(t_i)$ is the set of individuals who are still under study at the time just prior to t_i. The parameter estimates, $\hat{\beta}$, can be obtained by maximizing the partial likelihood (4.2.3).

Since the hazard function characterizes the survival function, the Cox PH model can be rewritten in terms of survival function:

$$S(t, X_n; \beta) = S_0(t)^{e^{X_n'\beta}},$$

where $S_0(t)$ denotes the baseline survival function given by $S_0(t) = exp\{-\int_0^t h_0(s)ds\}$. The baseline survival function estimate, $\hat{S}_0(t)$, can be obtained by the Kaplan-Meier

product limit estimate [56] or the Nelson-Aalen estimate [1]. Details of these estimators are presented in Appendix A. We use the Nelson-Aalen estimator to estimate the baseline survival function throughout this work.

Accelerated Failure Time (AFT) Model

Instead of modeling the hazard function as in the Cox PH model, the Accelerated Failure Time (AFT) model expresses the logarithm of the true survival time for the n^{th} individual as a linear regression model:

$$log(y_n) = \mu + X_n'\beta + \sigma U_n, \tag{4.2.4}$$

where, for the n^{th} individual, y_n is the true survival time, $X_n = (X_{n1}, \ldots, X_{np})$ are the covariates for the n^{th} individual, $\beta = (\beta_1, \ldots, \beta_p)$ are the regression coefficients, μ is a location parameter, and σ is a scale parameter. Here, U_n's are independent and identically distributed (i.i.d.).

Several works in the literature have explored semiparametric estimation of the coefficients in the AFT model with an unspecified error distribution. The least-squares method of Buckley-James [20] used the Kaplan-Meier estimator to adjust for the censored observations. Another popular method is the rank-based estimator using the score function of the partial likelihood (Ritov [88], Jin et al. [52]). Using the two-stage procedure, the semiparametric AFT model can be used in conjunction with dimension reduction methods. For instance, the method of PLS can be used in the first stage to reduce the dimension of the data. Adopting the modified version of PLS (Nguyen [80]), the semiparametric AFT model is used in the construction of the PLS weights to incorporate the censoring. In the second stage, the reduced data are fitted to a multivariate AFT model, where the coefficients are obtained

semiparametrically. Once the coefficients are estimated, the log of the lifetimes can be estimated. However, a drawback of the semiparametric approach is the difficulty in computing such estimators, even if there are only a few covariates (Jin et al. [52]).

This work focuses on the parametric AFT model where the distribution of the error is known. The AFT likelihood function for right-censored data is given by

$$L(\mu, \beta, \sigma) = \prod_{n=1}^{N} \left[\frac{1}{\sigma} f_0 \left(\frac{y_n - \mu - X_n'\beta}{\sigma} \right) \right]^{\delta_n} \left[S_0 \left(\frac{y_n - \mu - X_n'\beta}{\sigma} \right) \right]^{(1-\delta_n)} \tag{4.2.5}$$

where f_0 and S_0 denote the probability distribution function (pdf) and the cumulative distribution function (cdf) of the U_n's. Estimates for μ, β and σ are found by maximizing the likelihood function given in Eq. (4.2.5) (Klein and Moeschberger [59]).

As indicated in (Bedrick et al. [10]), if the baseline survival function S_0 is standard normal, then the log-normal survival model is obtained. If S_0 is logistic, then log-logistic model is obtained, and if S_0 is extreme-value, then we get the Weibull model. Details of the AFT model are provided in Appendix A.

Since the number of genes, p, far exceeds the number of cases N for microarray data, regression models such as the Cox PH and AFT models, cannot be applied directly. We first reduce the dimension of the gene expression data matrix from N x p to N x k, where $k \ll N \ll p$ via dimension reduction methods, and then apply the regression model in the reduced subspace. If we let $\tilde{X} = [\tilde{x}_1, \ldots, \tilde{x}_k]$ be the reduced data matrix after dimension reduction, then regression models such as the Cox and AFT models are applied to the reduced data matrix $\tilde{X}$.

As it is not possible to analytically compare the various approaches for dimension reduction for the various models discussed here, the next subsection discusses a Monte Carlo simulation that studies the performance of the various methods. Several criteria

for assessing the operating characteristics of the various methods are considered. The simulation procedure is discussed next.

4.2.2 Simulation Procedure

We compare the performance of rank-based modified versions of Partial Least Squares (PLS) to their unranked counterparts, and other well-known dimension reduction methods such as PCA, SIR, UNIV, SPCR and CPCR using the Cox and AFT regression models.

4.2.3 Simulation Setup

We used the simulation procedure described by Nguyen and Rocke [80], which is comprised of two main parts: generating gene expression values and generating the survival times. The details of the simulation procedure for the gene expression values and the survival times are provided in Appendix A.

The performance of the different dimension reduction methods are assessed in the presence of outliers in the response. In the simulations, the observed survival times are generated such that they have outliers for large values of p. Figure B.1 shows, for one simulation in the case $p = 100$ under the Cox model, the observed survival times $T_i = min(y_i, c_i)$ do not have outliers. However, for $p = 1000$, the T_i's have outliers. In these simulations, the effect of outliers in the response is investigated on the different dimension reduction methods (see Nguyen and Rojo [81, 82] for details).

Since the reduced data matrix is of dimension N x k after dimension reduction, two scenarios for the selection of k are considered for the different dimension reduction methods under the Cox model: 1) k is fixed across the different methods, and 2) k is selected based on the minimization of the cross-validation squared error of the

estimated survival function for each method. Under the AFT model, k is selected based on the minimized cross-validation of the squared error of fit. Details of fixing k under the Cox model are provided in Appendix A. We next describe the selection of k based on cross-validation.

4.2.4 Cross-validation (CV)

In practice, k is chosen by cross-validation, which leads to different k for different methods. In this work, we employ a 2−fold CV using the minimization of the squared error of the estimated survival function CV $(surv.error)$ under the Cox model, and the squared error of fit CV $(fit.error)$ under the AFT model, for the simulated data to compare the different methods. The $CV(surv.error)$ is defined as:

$$CV(surv.error) = \frac{1}{sM}\sum_{i=1}^{s}\sum_{m=1}^{M}\sum_{t\in D_m}\left[\hat{\bar{S}}_{-m}(t) - \hat{\bar{S}}_{m}(t)\right]^2,$$

where $i = 1, \ldots, s$ is the index for the simulation run, $s = 5000$ simulations, $m = 1, \ldots, M$ is the index for the fold, $M = 2$, D_m is the set of death times in the m^{th} fold, $\hat{\bar{S}}_m$ denotes the estimated survival function for the m^{th} fold, and $\hat{\bar{S}}_{-m}$ denotes the estimated survival function when the m^{th} fold is removed. In this setting, for each simulation run, we use a $50:50$ split of the data into a training set (index $-m$) and a test set (index m). Also, the estimated survival functions are evaluated using the covariates correponding to the individuals, i.e.,

$$\hat{\bar{S}}_m(t) = \frac{1}{N_m}\sum_{n=1}^{N_m}\hat{S}_{m,n}(t)$$

and

$$\hat{\bar{S}}_{-m}(t) = \frac{1}{N_{-m}}\sum_{n=1}^{N_{-m}}\hat{S}_{-m,n}(t)$$

where $N_m = N_{-m} = 25$ denotes the number of individuals either in test or training set, $\hat{S}_{m,n}$ is the estimated survival function for the n^{th} individual in the test set, and

$\hat{S}_{-m,n}$ is the estimated survival function for the n^{th} individual in the training set. The estimated survival function for the n^{th} individual are $\hat{S}_{m,n}(t) = \hat{S}_{0,m}(t)^{exp(X'_{m,n}\hat{\beta}_m)}$ and $\hat{S}_{-m,n}(t) = \hat{S}_{0,-m}(t)^{exp(X'_{-m,n}\hat{\beta}_{-m})}$.

Under the AFT model, since the logarithm of the lifetimes is represented as a linear regression model, it is natural to examine the squared error of fit. The $CV(fit.error)$ is defined as:

$$CV(fit.error) = \frac{1}{sM}\sum_{i=1}^{s}\sum_{m=1}^{M}\left[\frac{\sum_{l=1}^{N_m}\delta_{m,l}(i)\left(\hat{y}^*_{m,l}(i) - y^*_{m,l}(i)\right)^2}{\sum_{l=1}^{N_m}\delta_{m,l}(i)}\right]$$

where $i = 1, \ldots, s$ is the index for the simulation run, $s = 5000$ simulations, $m = 1, \ldots, M$ is the index for the cross-validation fold, $M = 2$, $l = 1, \ldots, N_m$ is the index for the individual in the m^{th} fold, $N_m = 25$ is the number of individuals in the m^{th} fold, and $\delta_{m,l}(i)$ denotes the censoring indicator for the l^{th} individual in the m^{th} fold of the i^{th} simulation. The $y^*_{m,l}(i)$'s are defined as

$$y^*_{m,l}(i) = log\left(y_{m,l}(i)\right)$$

where the $y_{m,l}(i)$'s are the actual lifetimes. The $\hat{y}^*_{m,l}(i)$ are the estimates of $y^*_{m,l}(i)$, and are given by

$$\hat{y}^*_{m,l}(i) = \hat{\mu}_{-m,AFT}(i) + \tilde{X}_{m,l}(i)'\hat{\beta}_{-m,AFT}(i)$$

where $\hat{\mu}_{-m,AFT}(i)$ and $\hat{\beta}_{-m,AFT}(i)$ are the coefficients estimated from the AFT model when the m^{th} fold is removed, and $\tilde{X}$ corresponds to the reduced data matrix after applying dimension reduction techniques.

For each method, a $CV(surv.error)$ for the Cox model or $CV(fit.error)$ for the AFT model is obtained for each value of λ, which is the tuning parameter for that method. Here, $\lambda < N_m$. In these simulations, we let $\lambda = 1, 2, \ldots, 20$. The optimal λ corresponds to the k that minimizes either $CV(surv.error)$ for the Cox model or $CV(fit.error)$ for the AFT model.

4.2.5 Performance Measures

For microarray data, it is important to select the relevant genes that relate to biological processes as well as accurately predicting the patients' survival (Cox model) or patients' lifetimes (AFT model). Thus, we are interested in assessing the performance of the different dimension reduction methods based on the mean squared error of the estimated coefficients on the genes, the mean squared error and bias of the estimated survival function, and the mean squared error of fit. In other words, once k is selected for each method, we compute the following measures.

The first measure that we examine is the $MSE(\beta)$ defined in terms of the weights placed on the genes

$$MSE(\beta) = \frac{1}{s}\sum_{i=1}^{s}\sum_{j=1}^{p}(\beta_j - \hat{\beta}_{ij})^2$$

where $i = 1, \ldots, s$ indicates the i^{th} simulation, and $j = 1, \ldots, p$ indicates the j^{th} gene. For the i^{th} simulation, the p x 1 vector $\hat{\beta}$ is obtained by $\hat{\beta} = W\hat{\beta}_R$ with W is the loadings or weights obtained from dimension reduction step (such as PCA, PLS,...), and $\hat{\beta}_R$ are the parameter estimates obtained from the Cox or AFT model.

The next two measures, $ave(d^2)$ and $ave(d^2.ind)$, are in terms of the mean squared error of the estimated survival function under the Cox model. The $ave(d^2)$ is defined as:

$$ave(d^2) = \frac{1}{s}\sum_{i=1}^{s}\sum_{t\in D_s}(\bar{S}_i(t) - \hat{\bar{S}}_i(t))^2$$

where for the i^{th} simulation, t corresponds to the observed death times, and for the Cox model,

$$\bar{S}_i(t) = S_0(t)^{exp(\bar{X}(i)'\beta)}$$

and

$$\hat{\bar{S}}_i(t) = \hat{S}_0(t)^{exp(\bar{X}(i)'\hat{\beta})}.$$

Here, both the true and estimated survival are obtained from the average of the covariates $\bar{X}$ in the i^{th} simulation, denoted by $\bar{X}(i)$, and $\hat{S}_0$ under the Cox model is the Nelson-Aalen estimator of the baseline survival function.

The next measure, $ave(d^2.ind)$, measures the mean squared error of survival where the survival function is evaluated using the covariates corresponding to the individuals, rather than the average of the covariates,

$$ave(d^2.ind) = \frac{1}{s}\frac{1}{N}\sum_{i=1}^{s}\sum_{n=1}^{N}\sum_{t\in D_s}(S_{in}(t) - \hat{S}_{in}(t))^2$$

where for the i^{th} simulation in the Cox model,

$$S_{in}(t) = S_0(t)^{exp(X_n(i)'\beta)}$$

and

$$\hat{S}_{in}(t) = \hat{S}_0(t)^{exp(X_n(i)'\hat{\beta})}$$

where $X_n(i)$ are the covariates corresponding to the n^{th} individual in the i^{th} simulation.

The next two measures, $ave(bias)$ and $ave(bias.ind)$ are in terms of bias of the estimated survival function under the Cox model. Both measures of bias are calculated at the deciles of the true survival function. The $ave(bias)$ is defined as

$$ave(bias) = \frac{1}{s}\sum_{i=1}^{s}\hat{\bar{S}}_i(t_q) - \bar{S}_i(t_q)$$

where $q = 0.1, 0.2, \ldots, 0.9$. In the Cox model, for the i^{th} simulation, $t_q = S_0^{-1}\left(q^{exp(-\bar{X}(i)'\beta)}\right)$ correspond to the deciles of the true survival function. In other words, $\bar{S}_i(t_q) = q$. The estimated survival is $\hat{\bar{S}}_i(t_q) = \left(\hat{S}_0(t_q)\right)^{exp(\bar{X}(i)'\hat{\beta})}$.

The $ave(bias.ind)$ is evaluated using the covariates corresponding to the individuals, rather than the average of the covariates,

$$ave(bias.ind) = \frac{1}{s}\frac{1}{N}\sum_{i=1}^{s}\sum_{n=1}^{N}\hat{S}_{in}(t_q) - S_{in}(t_q)$$

where for the i^{th} simulation and n^{th} individual, for the Cox model, $t_q = S_0^{-1}\left(q^{exp(-X_n(i)'\beta)}\right)$ so that $S_{in}(t_q) = q$, and $\hat{S}_{in}(t_q) = \left(\hat{S}_0(t_q)\right)^{exp(X_n(i)'\hat{\beta})}$.

The next measure, $MSE(fit)$, is the average of the squared residuals of the true lifetimes under the AFT model,

$$MSE(fit) = \frac{1}{s}\sum_{i=1}^{s}\left[\frac{\sum_{n=1}^{N}\delta_n(i)\left(\hat{y}_n^*(i) - y_n^*(i)\right)^2}{\sum_{n=1}^{N}\delta_n(i)}\right]$$

where for the i^{th} simulation and n^{th} individual,

$$y_n^*(i) = log\left(y_n(i)\right)$$

and

$$\hat{y}_n^*(i) = \hat{\mu}_{AFT}(i) + \tilde{X}_n(i)'\hat{\beta}_{AFT}(i)$$

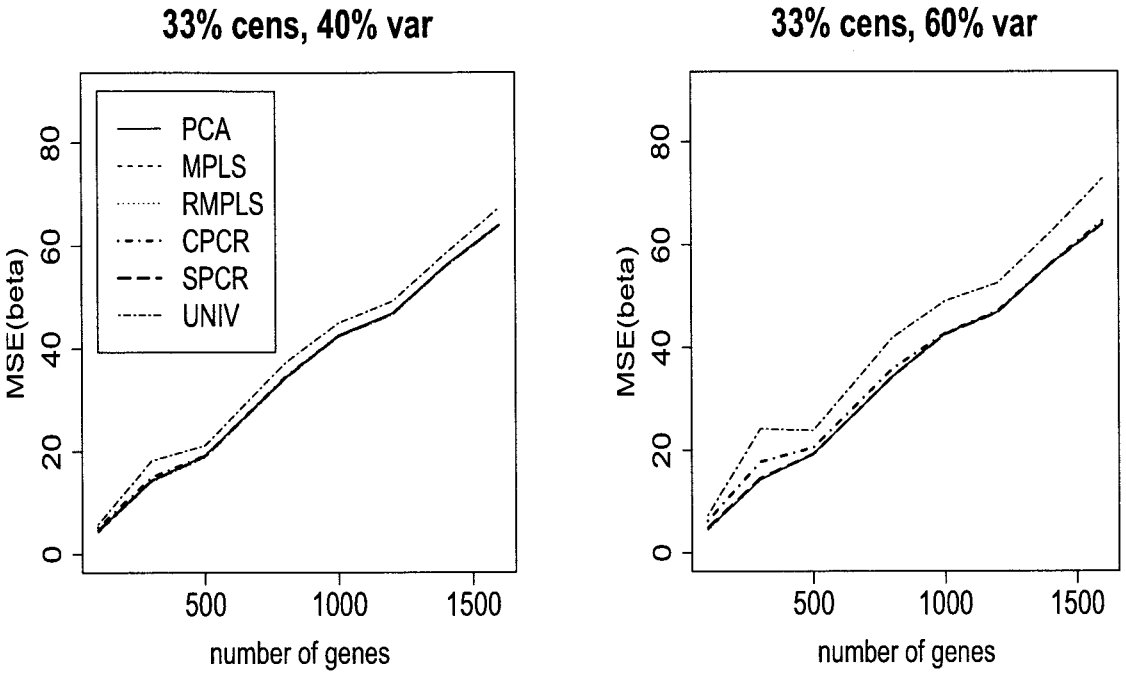
The simulation results for the various dimension reduction methods based on the above performance measures are provided next.

4.2.6 Simulation Results

Cox model

Figure 4.1 compares the $MSE(\beta)$ for PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV for censoring rate of 1/3 and total variation of predictor explained (TVPE) of 40% and 60%. In the case of $p = 100$ in the absence of outliers in the response, PCA, MPLS, RMPLS and SPCR perform relatively the same in terms of $MSE(\beta)$, and all four methods outperform CPCR and UNIV. In the case when $p \geq 300$ in the presence of outliers in the response, similar observations can be made as in the case of no outliers. We omit the plots for TVPE of 50% and 70% since they are similar to cases of 40% and 60%, and censoring rate of 1/2 since it is similar to the case of 1/3 censoring.

Figure 4.1 : Cox model: 1/3 censored. The mean squared error of the estimated weights on the genes, $MSE(\beta)$, for datasets with 40% and 60% TVPE accounted for by the first 3 PCs comparing PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV.



Figures 4.2 and 4.3 compare the $ave(d^2)$ of survival for PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV for censoring rate of 1/3 and 1/2, respectively, and TVPE of 40%, 50%, 60% and 70%. In the case when $p = 100$ in the absence of outliers in the response, RMPLS performs slightly better than MPLS, and both methods outperform PCA for low to moderate TVPE (40% and 50%). SPCR yields close $ave(d^2)$ to PCA, and all four methods RMPLS, MPLS, PCA and SPCR outperform both CPCR and UNIV. At high censoring rate of 1/2, the performance of all methods deteriorate because of the small effective sample size. However, the pattern remains the same as in the case of 1/3 censoring. This result is consistent with the findings

of Nguyen [79]. In the case when $p \geq 300$ in the presence of outliers, RMPLS substantially outperforms all other methods. MPLS is affected by outliers, since the method performs worse than PCA some of the times. SPCR performs better than PCA. UNIV performs surprisingly well, better than PCA in some instances. CPCR performs relatively worst among all the methods.

Figure 4.4 compares the $ave(d^2.ind)$ of survival for PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV for censoring rate of 1/3, and TVPE of 40%, 50%, 60% and 70%. In the case when $p = 100$ in the absence of outliers in the response, RMPLS performs slightly worse than MPLS. Both methods outperform all other methods for all TVPE. Again, similar to the results for the measure $ave(d^2)$, SPCR yields close $ave(d^2.ind)$ to PCA, and both methods perform better than CPCR. UNIV performs worst among all the considered methods. In the case when $p \geq 300$ in the presence of outliers, RMPLS substantially outperforms all other methods. Again, MPLS is affected by outliers, since the method performs worse than SPCR most of the times. Both SPCR and MPLS outperform PCA. UNIV performs well, better than PCA in some instances. CPCR generally performs worst among all the methods. The results for censoring rate of 1/2 are similar to those for censoring rate of 1/3 (not shown), although the performance of the methods deteriorate due to a high censoring rate.

Figure B.2 compares the $ave(bias)$ of the estimated survival function for PCA, MPLS, RMPLS, SPCR, CPCR and UNIV for censoring rate of 1/3, $p = 100$, 500 and 800, and TVPE of 50%, 60% and 70%. The results for the cases $p = 300$, 1000, 1200, 1400 and 1600 are similar to the results for $p = 500$, and 800, so we omit these plots. Also, the results for the censoring rate of 1/2 are not shown since they are similar to the results for censoring rate of 1/3. However, at high censoring rate of 1/2, the performance of all methods deteriorate because of the small effective sample

Figure 4.2 : Cox model: 1/3 censored. The mean squared error of the survival function evaluated at the average of the covariates, $ave(d^2)$ of survival, for datasets with 40%, 50%, 60% and 70% TVPE accounted for by the first 3 PCs comparing PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV. The $x-$axis denotes the number of genes, p, and the y-axis denotes the $ave(d^2)$ of survival.

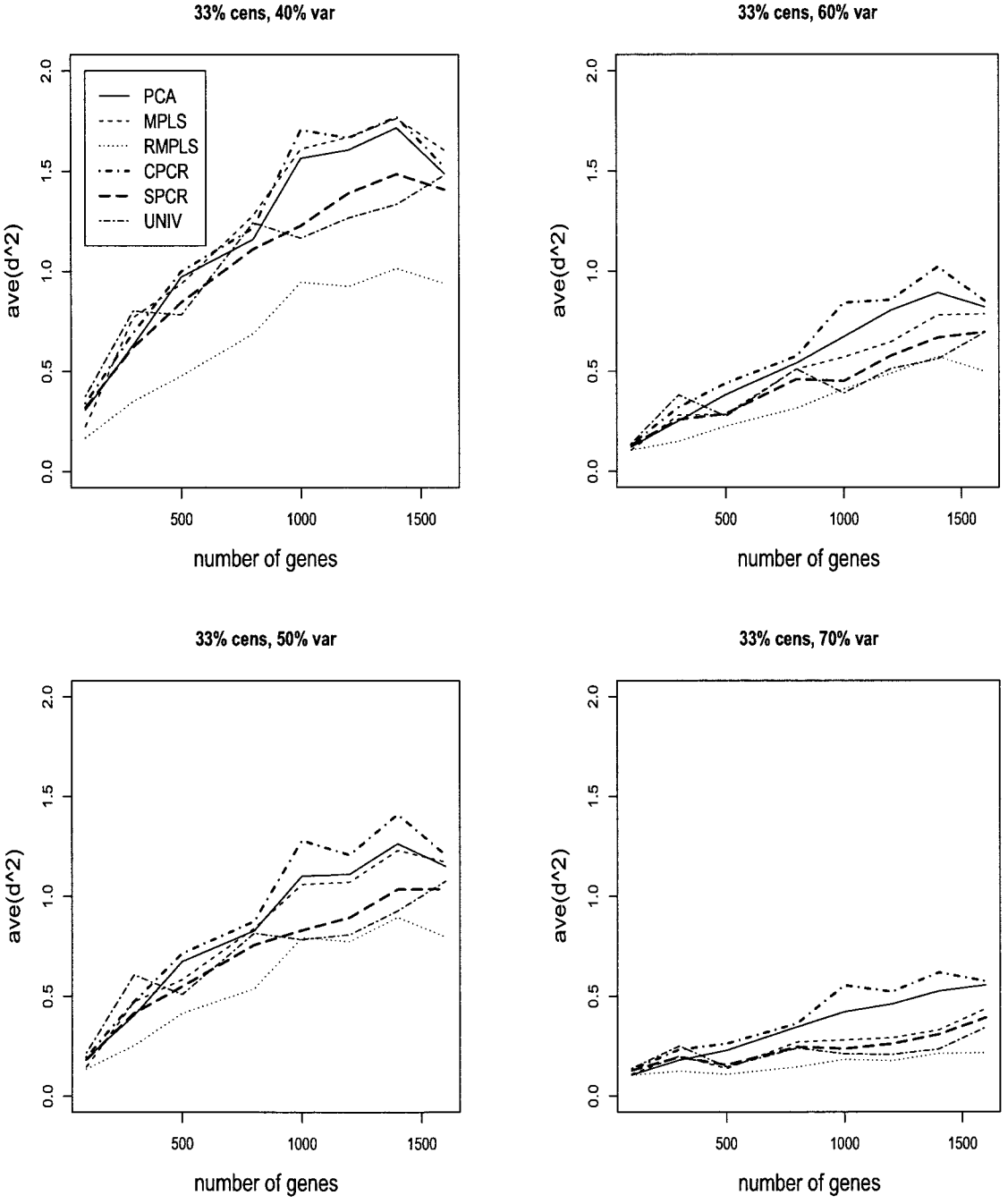


Figure 4.3 : Cox model: 1/2 censored. The mean squared error of the estimated survival function evaluated at the average of the covariates, $ave(d^2)$ of survival, is plotted for datasets with 40%, 50%, 60% and 70% TVPE accounted for by the first 3 PCs comparing PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV.

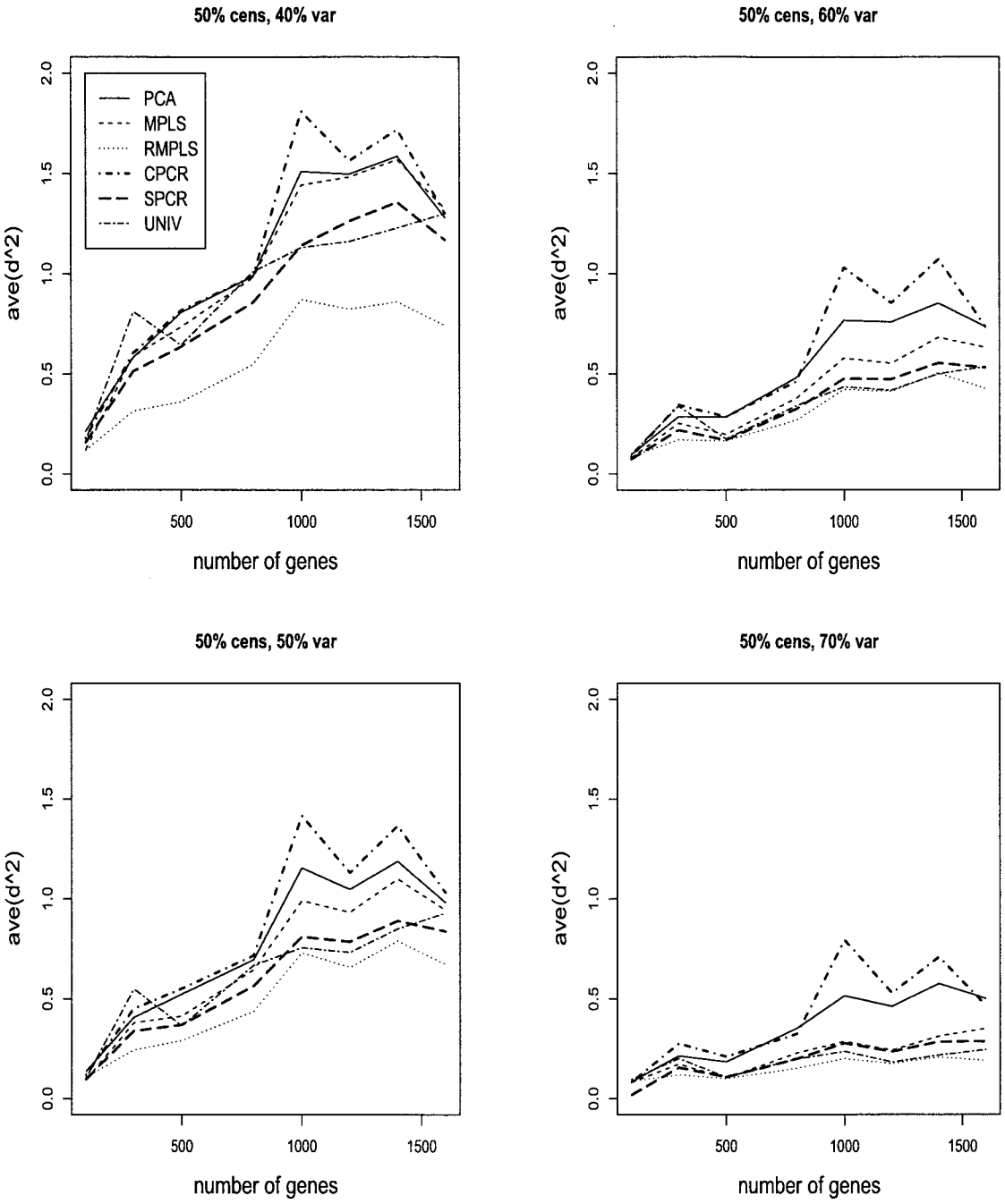
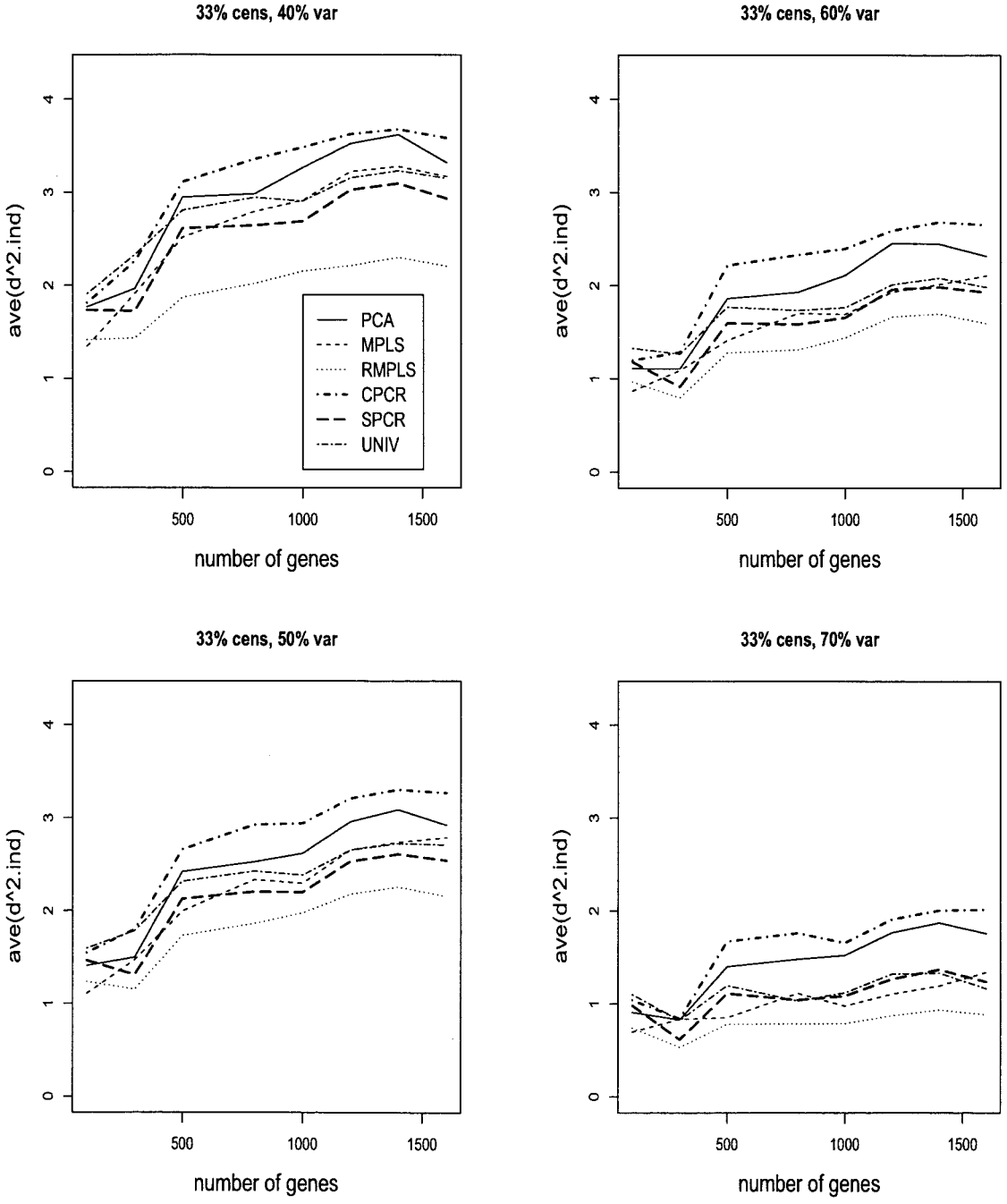


Figure 4.4 : Cox model: 1/3 censored. The mean squared error of the estimated survival function evaluated at the covariates of the individuals, $ave(d^2.ind)$ of survival, is plotted for datasets with 40%, 50%, 60% and 70% TVPE accounted for by the first 3 PCs comparing PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV. The $x-$axis denotes the number of genes, p, and the y-axis denotes $ave(d^2.ind)$ of survival.



size. RMPLS generally outperforms all other methods, including MPLS, for small to medium deciles ($q = 0.1, \ldots, .5$) in both cases when $p = 100$ in the absence of outliers in the response and when $p \geq 300$ in the presence of outliers. For large deciles ($q = .6, \ldots, .9$), there is no clear-cut winner among the methods. In the case when $p = 100$ in the absence of outliers in the response, both RMPLS and MPLS outperform PCA for all deciles ($q = .1, \ldots, .9$). SPCR and CPCR yield close estimates to PCA for the case of 1/3 censoring, and UNIV performs relatively worst. In the case when $p \geq 300$ in the presence of outliers in the response, MPLS is affected by outliers, since the method performs worse than PCA, SPCR, and UNIV some of the times.

Figure B.3 compares the $ave(bias.ind)$ of the estimated survival function for PCA, MPLS, RMPLS, SPCR, CPCR and UNIV for censoring rate of 1/3, $p = 100, 500$ and 800, and TVPE of 50%, 60% and 70%. In the case when $p = 100$ in the absence of outliers in the response, RMPLS is comparable to MPLS. Both methods outperform all other methods, including PCA, for all TVPE for small to medium deciles ($q = 0.1, \ldots, .5$). Also, SPCR and UNIV perform slightly better than PCA and CPCR. In the case when $p \geq 300$ in the presence of outliers in the response, RMPLS outperforms all other methods, including MPLS, for $q = 0.1, \ldots, .5$. For large deciles $q = 0.6, \ldots, .9$, RMPLS, MPLS, SPCR and UNIV perform relatively the same. Furthermore, RMPLS, SPCR, and UNIV perform slightly better than PCA and CPCR for all deciles.

Figure B.4 compares the $MSE(\beta)$, $ave(d^2)$, and $ave(d^2.ind)$ for methods coupled with SIR (PCA and MPLS) and their un-SIR counterparts for censoring rate of 1/3 and TVPE of 50% and 70% using the baseline exponential survival in the Cox model. SIR does not improve upon the performance of the dimension reduction methods.

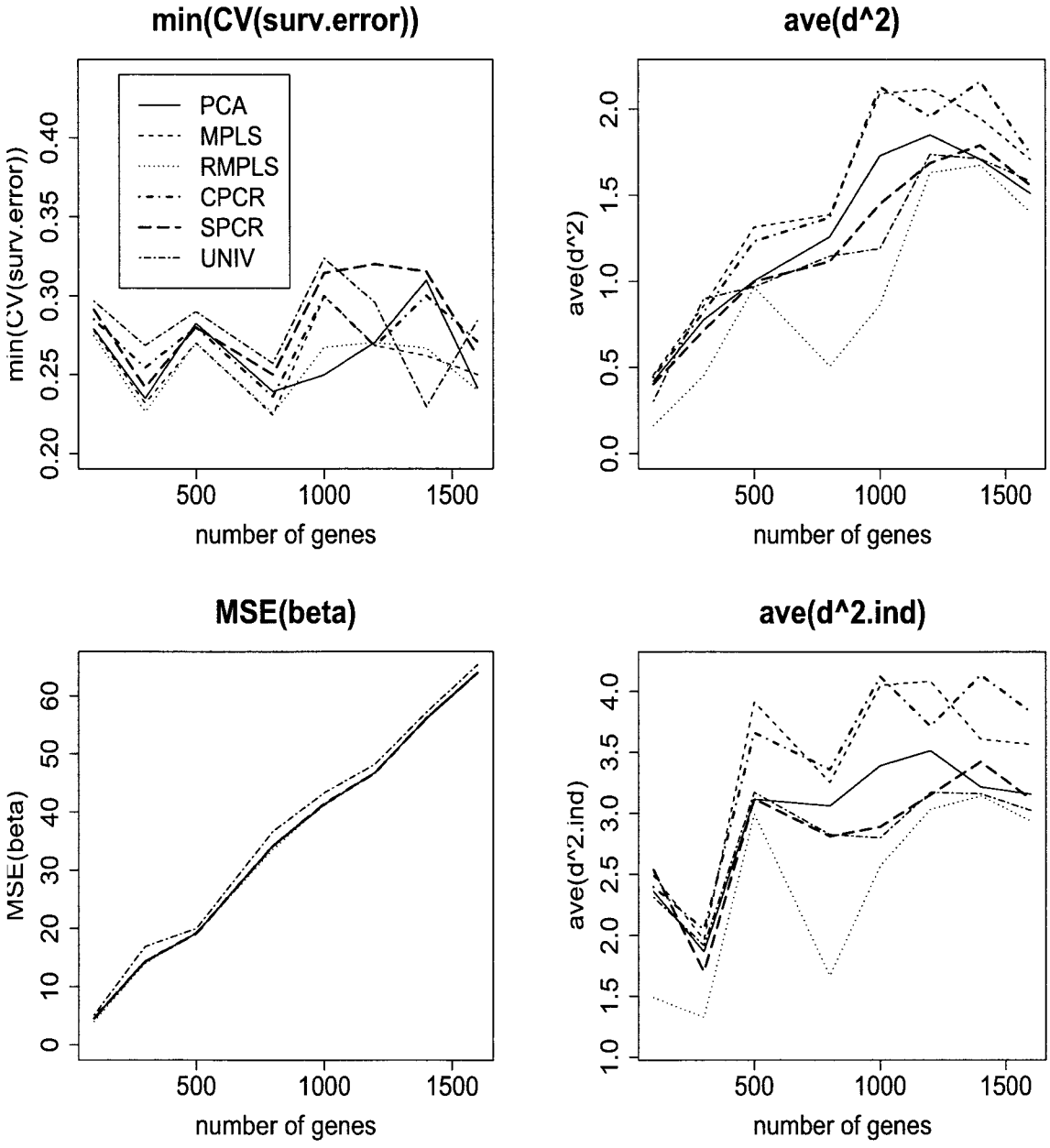
The results are similar for TVPE of 40% and 60%, censoring rate of 1/2, and the two bias measures ($ave(bias)$ and $ave(bias.ind)$), so we omit these plots.

We should observe that all methods generally improve as the TVPE increases for the two measures of mean squared error of the estimated survival function and the two measures of bias.

Under the Cox model, using 2−fold CV based on the minimization of the squared error of the estimated survival function for each method, k is selected. Once the CV is performed, we can use k with the simulated data as before, and obtain the mean square error for the β's and the estimated survival function. Figure 4.5 compares the $CV(surv.error)$, $MSE(\beta)$, $ave(d^2)$ and $ave(d^2.ind)$ among PCA, MPLS, RMPLS, CPCR, SPCR and UNIV. RMPLS generally outperforms other methods in terms of $CV(surv.error)$, $ave(d^2)$ and $ave(d^2.ind)$ for both cases when outliers are present and absent in the response. MPLS is affected by outliers, since the method performs worse than PCA in terms of $ave(d^2)$ and $ave(d^2.ind)$. In terms of $MSE(\beta)$, PCA, MPLS, RMPLS, CPCR and SPCR perform relatively the same, and they all outperform UNIV. Using CV, RMPLS is also better variant of PLS than MPLS as in the case when the number of components, k, is fixed for all the methods.

Table C.5 compares the mean squared error of the estimated weights on the genes $MSE(\beta)$ and the mean squared error of the estimated survival function using the individual covariates $ave(ds.ind)$ for two procedures: 1) using PCA or RMPLS directly, and 2) combining RP and PCA or RMPLS. Procedure 2 yields similar results to those of procedure 1. Thus, RP can be combined with other dimension reduction methods without losing much of the information in the original data.

Figure 4.5 : Cox model: 1/3 censored. k is chosen by cross-validation (CV). The minimized CV of the squared error of the estimated survival function $min(CV(surv.error))$, mean squared error of the estimated weights on the genes $MSE(\beta)$, mean squared error of the estimated survival function evaluated at the average of the covariates $ave(d^2)$, and mean squared error of the estimated survival function evaluated at the covariates of the individuals $ave(d^2.ind)$ comparing PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV based on 1000 simulations are plotted.



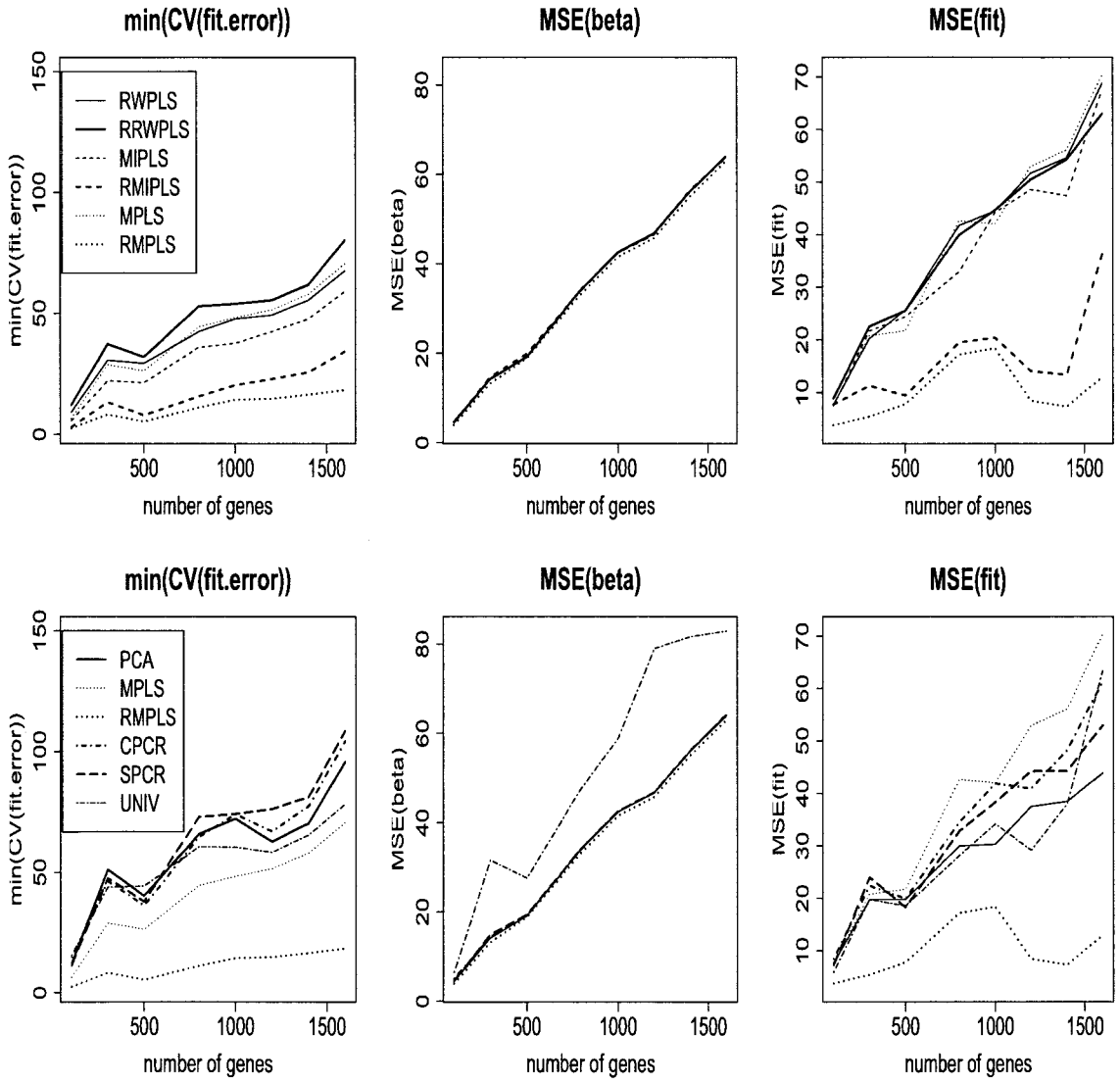
AFT model

Figure 4.6 compares the $CV(fit.error)$, $MSE(\beta)$, and $MSE(fit)$ for RWPLS, RRWPLS, MIPLS, RMIPLS, MPLS, and RMPLS for censoring rate of 1/3 under the AFT exponential model. In terms of $MSE(\beta)$, the ranked versions of PLS are comparable to their un-ranked counterparts. RMPLS outperforms other methods, including MPLS, in terms of $CV(fit.error)$ and $MSE(fit)$ for both cases when outliers are absent ($p = 100$) and present ($p \geq 300$) in the response. In the absence of outliers in the response ($p = 100$), the ranked versions of RWPLS and MIPLS are comparable to their un-ranked counterparts in terms of $CV(fit.error)$ and $MSE(fit)$. RMPLS and RMIPLS significantly improve their un-ranked counterparts in terms of $CV(fit.error)$ and $MSE(fit)$ in the presence of outliers, while RRWPLS does not necessarily outperform its un-ranked version. Similar results are obtained for the lognormal mixture model (Figure B.7), lognormal model (Figure B.8) and log-t model (Figure B.9).

Figure 4.6 (bottom row) compares the $CV(fit.error)$, $MSE(\beta)$, and $MSE(fit)$ for PCA, MPLS, RMPLS, CPCR, SPCR, and UNIV for censoring rate of 1/3 under the AFT exponential model. In terms of $MSE(\beta)$, PCA, MPLS, RMPLS, CPCR and SPCR perform relatively the same when outliers are absent ($p = 100$) and present ($p \geq 300$) in the response. UNIV performs worst among the methods in terms of $MSE(\beta)$. RMPLS outperforms all other methods in terms of $CV(fit.error)$ and $MSE(fit)$ in the presence of outliers in the response. Similar results are obtained for the lognormal mixture model (Figure B.7), lognormal model (Figure B.8) and log-t model (Figure B.9).

Since simulation results of the assessment of the different dimension reduction methods do not translate to similar findings in real microarray datasets, we also

Figure 4.6 : AFT exponential model: 1/3 censored. k is chosen by cross-validation (CV). The minimized CV of the fit error $min(CV(fit.error))$, mean squared error of the estimated weights on the genes $MSE(\beta)$, and mean squared error of fit $MSE(fit)$ comparing RWPLS, RRWPLS, MIPLS, RMIPLS, MPLS, and RMPLS (top row), and comparing PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV (bottom row) based on 5000 simulations are plotted.



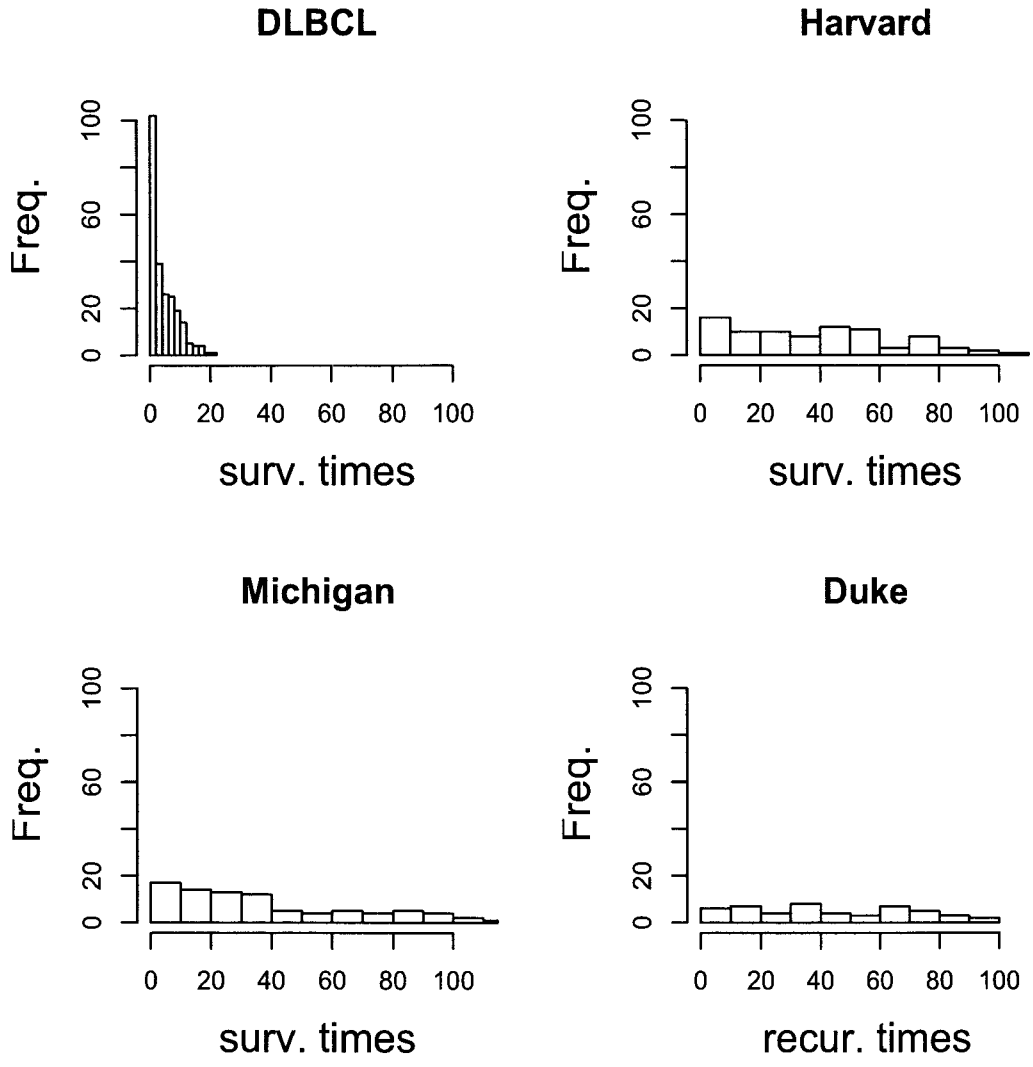
assess the performance of the methods based on four real datasets as provided in the next subsection. Since the reduced data matrix (after dimension reduction) is of dimension p x k, the selection of k is based on cross-validation.

4.2.7 Real Datasets

The first dataset is the Diffuse Large-B-cell Lymphoma (DLBCL) data described in Rosenwald et al. [90], and Bair and Tibshirani [8]. There are 240 patients, 7399 genes, and 42.5% of the patient survival times are censored. Five of the survival times are 0, so we set these survival times to 0.001 in order to apply the AFT model. The Harvard Lung Carcinoma dataset consists of 84 cases, 12625 genes, and 42.9% of the cases are censored (Bhattacharjee et al. [15]). The Michigan Lung Adenocarcinoma consists of 86 cases, 7129 genes, and 72.1% of the cases are censored (Beer et al. [11]). The Duke Breast Cancer dataset consists of 49 cases, 7129 genes, and 69.4% of the cases are censored (West et al. [96]). Note that the survival times in the Harvard, Michigan and Duke datasets have longer tails than those of the DLBCL dataset (Figure 4.7). We used a 3−fold CV for the four datasets: 80 samples in test set and 160 in the training set for the DLBCL data, 28 in test set and 56 in the training set for the Harvard data, 28 in test set and 58 in the training set for the Michigan data, and 16 in test set and 33 in the training set for the Duke data. For the Harvard data, we first screened out the genes using UNIV under an AFT model to retain 7189 top-ranked genes. The cross-validation is based on 1000 repetitions. The comparison of the different dimension reduction methods is based on the minimized $CV(surv.error)$ for the Cox model and $CV(fit.error)$ for the AFT model.

Table 4.1 shows the minimized $CV(surv.error)$ and the standard error of the 1000 repeated runs for the various methods under the Cox model for the DLBCL and

Figure 4.7 : Histograms of the survival times for DLBCL, Harvard, Michigan and Duke datasets. The survival times for the Harvard, Michigan and Duke datasets have longer tails than the survival times for the DLBCL dataset.



Harvard datasets. RMPLS outperforms all other methods for the Harvard data, in the presence of outliers in the response. Also, the method is comparable to MPLS and other methods for the DLBCL data in the absence of outliers.

Table 4.1 : Cox model: DLBCL and Harvard datasets. k chosen by CV for the different methods. The minimized cross-validation of the squared error of the estimated survival function $min(CV(surv.error))$, and its standard error of the 1000 repeated runs are shown.

	DLBCL			HARVARD		
Method	**k**	*error*	*SE*	**k**	*error*	*SE*
PCA	7	0.1026	0.0336	13	0.121	0.06
MPLS	1	0.1074	0.0372	1	0.1304	0.0654
RMPLS	1	0.1056	0.0354	1	0.1124	0.0305
CPCR	2	0.1014	0.0346	2	0.1402	0.0727
SPCR	1	0.1063	0.0353	3	0.1473	0.0822
UNIV	11	0.1221	0.0383	14	0.1663	0.0863

Tables 4.2, 4.3, C.2, and C.3 show the minimized $CV(fit.error)$ and the standard error of the 1000 repeated runs for the different methods under the AFT exponential, lognormal mixture model, lognormal and log-t models, respectively. Under the lognormal mixture model, RMPLS outperforms other methods. Under the exponential model, RMPLS generally outperforms other methods, except for the DLBCL (short tail) and Harvard datasets in which the method is slightly outperformed by RMIPLS. The standard error for the minimized $CV(fit.error)$ over the 1000 repeated runs for

RMPLS is comparable to other variants of PLS. Results for the lognormal and log-t models are similar to the results for the lognormal mixture model.

We also explored the similarity between MPLS and RMPLS (Cox model and AFT model), RWPLS and RRWPLS, and MIPLS and RMIPLS in the ranking of the significant genes based on the absolute value of the estimated weights on the genes (AEW), where AEW is defined as,

$$AEW = |W\hat{\beta}_R^*|$$

where W is the matrix of the weights obtained from the dimension reduction step for MPLS or RMPLS using the whole datasets, and $\hat{\beta}_R^* = \frac{\hat{\beta}_R}{se(\hat{\beta}_R)}$, where $se(\hat{\beta}_R)$ denotes the standard error of the estimate $\hat{\beta}_R$. Here R denotes either the Cox or AFT regression model. Table C.1 shows the number of top-ranked genes in common between MPLS and RMPLS out of k considered top-ranked genes for the two datasets using only the first component under the Cox model. We should observe that MPLS and RMPLS select many genes that are in common. Since the response of the Harvard dataset has outlying observations, the number of common genes selected by the two methods is generally less than that of the DLBCL dataset in the absence of outliers. Table C.4 shows the number of top-ranked genes in common between MPLS and RMPLS, RWPLS and RRWPLS, and MIPLS and RMIPLS, out of k considered top-ranked genes for the two datasets using only the first component under the AFT lognormal mixture model. MPLS and RMPLS, and MIPLS and RMIPLS select many of the genes in common. Again, the number of common genes selected by the ranked versions of PLS and their un-ranked counterparts for the Harvard, Michigan and Duke datasets is generally less than that of the DLBCL dataset because the survival times of the Harvard, Michigan and Duke datasets have longer tails than those of the DLBCL dataset.

Table 4.2 : AFT exponential model: DLBCL, Harvard, Michigan and Duke datasets. k chosen by CV for the different methods. The minimized cross-validation of the squared fit error $min(CV(fit.error))$, and its standard error of the 1000 repeated runs are shown.

	DLBCL			**HARVARD**			**MICHIGAN**			**DUKE**		
Method	k	error	SE	k	error	SE	k	error	SE	k	error	SE
PCA	7	5.9975	0.9198	3	3.5506	0.6048	5	6.8389	1.4619	5	20.2349	5.7284
MPLS	3	2.9528	0.5285	2	1.2639	0.4307	3	3.1026	1.0491	1	12.4282	3.7951
RMPLS	3	2.4344	0.434	1	1.1679	0.2532	3	2.3919	0.7124	2	5.8008	2.4583
RWPLS	1	6.2211	0.8641	1	2.5704	0.2964	2	4.3526	0.9748	1	11.6507	2.7335
RRWPLS	1	5.7951	0.7833	1	2.6999	0.5653	2	3.2623	1.2633	2	6.9515	2.6657
MIPLS	3	3.1403	0.4513	2	1.1875	0.6105	2	3.1731	1.5248	2	11.8239	4.9705
RMIPLS	5	2.2963	0.4302	1	1.1585	0.2791	1	2.4789	1.5366	1	8.409	2.9436
CPCR	1	6.6413	1.1556	1	3.48	0.9098	1	7.2268	1.601	2	17.4874	5.3907
SPCR	1	6.3594	0.9723	1	3.8267	1.4767	1	10.6534	2.047	2	18.9023	5.5066
UNIV	9	6.5043	1.0576	9	2.9802	1.0017	5	7.1902	2.6427	4	17.5441	5.5117

Table 4.3 : AFT Lognormal Mixture model: DLBCL, Harvard, Michigan and Duke datasets. k chosen by CV for the different methods. The minimized cross-validation of the squared fit error $min(CV(fit.error))$, and its standard error of the 1000 repeated runs are shown.

	DLBCL			**HARVARD**			**MICHIGAN**			**DUKE**		
Method	k	error	SE	k	error	SE	k	error	SE	K	error	SE
PCA	5	5.3338	0.6355	6	1.4313	0.2179	4	4.1421	0.8324	4	19.1543	6.5068
MPLS	3	2.5578	0.4013	1	0.5081	0.1749	3	1.0418	0.4578	1	15.9075	4.6314
RMPLS	5	1.619	0.3016	2	0.3644	0.1295	4	0.6373	0.2462	3	5.4052	3.1271
RWPLS	1	5.5406	0.6855	1	1.3515	0.2165	1	4.9017	0.8752	1	13.1164	2.4758
RRWPLS	1	5.3157	0.7607	1	2.0553	0.3194	1	4.073	0.7727	2	9.5983	3.4348
MIPLS	3	2.795	0.43	2	0.5784	0.1639	2	1.7188	0.4036	2	11.7866	3.9348
RMIPLS	4	1.9231	0.638	2	0.5573	0.1549	2	1.1113	0.7714	1	10.4242	3.9714
CPCR	7	4.5904	0.8113	5	1.1098	0.2223	4	2.4685	0.5096	4	9.8773	3.6931
SPCR	1	5.4759	0.6712	1	2.1183	0.349	2	5.099	1.0204	2	22.1386	7.1154
UNIV	5	4.0944	0.7591	6	0.8381	0.38	6	1.7173	0.4515	7	10.8	4.0594

4.2.8 Discussion and Extensions

The simulation study indicates that the Rank-based Modified Partial Least Squares (RMPLS) outperforms the other considered methods in the presence of outliers in the response, and is competitive to MPLS and PCA in the absence of outliers. By using the rank-based approach, MPLS is improved in the presence of outliers.

There are several limitations to our simulation study. In these simulations, the gene expression levels x_{ij}'s are taken to be $x_{ij} = exp(x^*_{ij})$, where the x^*_{ij}'s are composed of linear combinations of d underlying components, each normally distributed with a certain mean and variance, and an error component, normally distributed with a different mean and variance. We should observe that the linear combination of the d underlying components is also normally distributed, and thus, x^*_{ij} is only composed of an underlying component and an error component. By having d underlying components, we have to take into account the weights for these components, r_{ki}, for $k = 1, \ldots, d$, and $i = 1, \ldots, N$. The survival and censoring times depend on the gene expressions, which in turn depend on the r_{ki}. A poor choice of the weights would make some of the observed survival times $T_i = min(y_i, c_i)$ outliers. For example, if we take $r_{ki} \sim Exp(10)$ in the Cox model, then the response has outliers for both cases $p = 100$ and $p = 1000$ as seen in Figure B.5. Figure B.6 compares the $MSE(\beta)$, $ave(d^2)$, $ave(d^2.ind)$, $ave(bias)$ for $p = 100$, and $ave(bias.ind)$ for $p = 100$, for the case $r_{ki} \sim Exp(10)$ of PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV for censoring rate of 1/3. In terms of mean squared error of the estimated survival function ($ave(d^2)$ and $ave(d^2.ind)$), RMPLS outperforms all other methods, including MPLS. Also, RMPLS outperforms all other methods for small to medium deciles ($q = .1, \ldots, .5$) in terms of the bias of the estimated survival function ($ave(bias)$ and $ave(bias.ind)$) in the case $p = 100$. Similar results were obtained for $p = 300$, 500, 800, 1000,

1200, 1400 and 1600. Also, a similar pattern is observed if $r_{ki} \sim Uniform(0, 0.5)$ or $r_{ki} \sim N(0, 0.25^2)$. For a detailed discussion, see Nguyen and Rojo [81, 82].

Furthermore, the magnitude of the β's, the coefficients for the genes, and hence, the survival times, are controlled by the variance σ_π^2. In these simulations, we fix $\sigma_\pi = 0.2$, so that we have outliers in the response for large values of p. However, we can vary σ_π as we increase p so that the survival times do not have outliers. The results (not included in this work) indicate that the performance of the dimension reduction methods for large values of p are similar to that in the case $p = 100$ in the absence of outliers for $r_{ki} \sim Unif(-0.2, 0.2)$ (see Nguyen and Rojo [81, 82] for details).

Chapter 5

Conclusions

In this chapter, we discuss the results of the dimension reduction methods tested in this work. We first summarize the results of our improvements on the lower bound for k for the method of Random Projection in section 5.1, followed by the results of our proposed method of Rank-based Modified Partial Least Squares (RMPLS) in section 5.2.

5.1 Summary of the Improvements on the Lower Bound for k for the JL Lemma

A computational simple method of dimension reduction that has attracted a lot of attention lately is Random Projection (RP). The method projects the original N **x** p data matrix X onto a lower k-dimensional subspace using a p **x** k random projection matrix Γ. The motivation of RP is the classic result of the Johnson-Lindenstrauss (JL) Lemma [54], which states that a set of N points in p-dimensional space can be projected onto a $k = O(\ln N/\epsilon^2)$-dimensional subspace such that the pairwise distance among the points is preserved within a factor of $1 \pm \epsilon$. Several improvements on the proof of the original JL Lemma as well as on the lower bound for k have been reported over the years. Among the reported improvements, Dasgupta and Gupta [28] provided a smallest known lower bound for k with the Gaussian random projection matrix using the moment generating function (mgf) approach. In this

work, we revisited the JL Lemma, and provided an improvement on the Dasgupta and Gupta bound by 1) using the moment generating function technique, and 2) working directly with the distribution of the random Euclidean distances. An improvement of $11 - 34\%$ on the Dasgupta and Gupta bound is obtained with approach 2.

Achlioptas [3] provided the same lower bound for k as Dasgupta and Gupta using a projection matrix consisting of i.i.d. entries drawn from distribution given in Eq. (3.4.1) ($q = 1$ or $q = 3$). In this work, we provided an improvement to the Achlioptas bound for the Rademacher random matrix ($q = 1$) by taking advantage of the properties of the Rademacher random variable. In particular, we provided an improvement to the Achlioptas bound using 1) Hoeffding's Inequality based on the mgf approach, 2) Berry-Esseen Theorem, and 3) Pinelis Inequality. Improvements of 15% and $10 - 40\%$ are obtained with approaches 2 and 3 for $\epsilon = 0.1$, respectively. We also provide an alternate proof to the Achlioptas Theorem, and discuss the case for asymmetric simple random matrices in this work.

Results in the literature state that the JL Lemma cannot be extended to the L_1 norm (L_1-L_1 projection). However, for the L_2-L_1 random projection, Matousek [75] obtained a lower bound for k using a sparse Gaussian and Achlioptas random matrices. In this work, we provided an improvement to the Matousek bound of $36 - 40\%$ using the mgf approach with the Gaussian and Achlioptas random matrices.

5.2 Summary of Rank-based Modified Partial Least Squares

A popular dimension reduction method that incorporates both the covariates and the response is Partial Least Squares (PLS). However, the usual Pearson covariance/correlation measure in the optimization criterion of PLS is influenced by outliers. In this work, we have proposed to replace the Pearson correlation measure with the

Spearman rank correlation measure. This variant of PLS is denoted as Rank-based Modified Partial Least Squares (RMPLS). RMPLS is insensitive to outlying values in both the covariates and response, and also incorporates the censoring information. The weight vectors for RMPLS are derived as solutions to an optimization criterion, and the algorithm to RMPLS is also provided. Simulation results as well as results for real datasets under the Cox PH and AFT models indicate that RMPLS outperforms other considered methods when outliers are present in the response, and is competitive to other methods including the regular PLS in the absence of outliers.

Bibliography

[1] Aalen, O. O. A linear regression model for the analysis of lifetimes. *Statistical Medicine*, **8**, 907–925, 1989.

[2] Access Excellence Resource Center. `www.accessexcellence.org`.

[3] Achlioptas, D. Database-friendly random projections. *Proc. ACM Symp. on the principles of database systems*, 274–281, 2001.

[4] Ailon, N., and Chazelle, B. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. *Proc. 38th ACM Symp. Theory of Computing*, 557–563, 2006.

[5] Ailon, N., and Liberty, E. Fast dimension reduction using Rademacher series on dual BCH codes. In *Symp. on Discrete Algorithms*, 1–9. San Francisco, CA, 2008.

[6] Ailon, N., Liberty, E., and Singer A. Dense Fast Random Projections and Lean Walsh Transforms. In *Proc. of 11th and 12th International Workshop on Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques*, 512–522. Springer-Verlag, 2008.

[7] Arriaga, R. I., and Vempala, S. An algorithmic theory of learning: robust concepts and random projections. *40th Annual Symposium on Foundations of Computer Science*. New York, NY, 1999.

[8] Bair, E., and Tibshirani, R. Semi-supervised methods to predict patient survival from gene expression data. *PLoS Biology*, **2**, 511–522, 2004.

[9] Bair, E., Hastie, T., and Tibshirani, R. Prediction by supervised principal components. *Journal of American Statistical Association*, **101**, 119–137, 2006.

[10] Bedrick, E. J., Exuzides, A., Johnson, W. O., and Thurmond, M. C. Predictive influence in the accelerated failure time model. *Biostatistics*, **3.3**, 331–346, 2002.

[11] Beer D. G., Kardia S. L. R., Huang C., Giordano, T. J., Levin, A. M., Misek, D. E., Lin, L., Chen, G., Gharib, T. G., Thomas, D. G., Lizyness, M. L., Kuick, R., Hayasaka, S., Taylor J. M. G., Iannettoni, M. D., Orringer, M. B., Hanash, S., Gene-expression profiles predict survival of patients with lung adenocarcinoma, *Nature Medicine* **8.8**, doi: 10.1038/nm733, 2002.

[12] Bertoni, A., and Valentini, G. Random projections for assessing gene expression cluster stability. In *IJCNN 2005, the IEEE-INNS International Joint Conference on Neural Networks*. Montreal, 2005.

[13] Bertoni, A., and Valentini, G. Ensembles based on random projections to improve the accuracy of clustering algorithms. `http://eprints.pascal-network.org/archive/00002362/01/bertoni-vale-WIRN05.pdf`. Milano, 2006.

[14] Bertoni, A., Valentini, G., Folgieri, R., and Piuri, V. Ensembles based on random projections for gene expression data analysis. `http://www.mtcube.com/Tesi-Folgieri.pdf`. Archivio Istituzionale della Ricerca, Milano, 2008.

[15] Bhattacharjee A., Richards W. G., Staunton J., Li, C., Monti, S., Vasa, P., Ladd, C., Beheshti, J., Bueno, R., Gillette, M., Loda, M., Weber, G., Mark,

E. J., Lander, E. S., Wong, W., Johnson, B. E., Golub, T. R., Sugarbaker, D. J., Meyerson, M., Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses, *PNAS* **98.24**, 13790–13795, 2001.

[16] Bingham, E. and Mannila, H. Random projection in dimensionality reduction: applications to image and text data. In *Proc. of KDD*, pp. 245–250, San Francisco, CA, 2001.

[17] Boulesteix, A., and Strimmer, K. Partial Least Squares: a versatile tool for the analysis of high-dimensional genomic data. *Briefings in Bioinformatics*, **8.1**, 32–44, 2006.

[18] Bovelstad, H. M., Nygard, S., Storvold, H. L., Aldrin, M., Borgan, O., Frigessi, A., and Lingjaerde, O. C. Predicting survival from microarray data - a comparative study. *Bioinformatics Advanced Access* 2007.

[19] Brinkman, B. and Charikar, M. On the impossibility of dimension reduction in L_1. *Proc. 44th IEEE Symp Foundations of Computer Science*, 514–523, 2003.

[20] Buckley, J. and James, L. Linear regression with censored data. *Biometrika*, **66**, 429–436, 1979.

[21] Bura E. and Pfeiffer, R. M. Graphical methods for class prediction using dimension reduction techniques on DNA microarray data, *Bioinformatics*, **19**, 1252–1258, 2003.

[22] Candes E. J., and Tao, T. Near-optimal signal recovery from random projections: universal encoding strategies?. *Information Theory, IEEE Transactions on*, **52.12**, 5406–5425, 2006.

[23] Cattell R. B. The scree test for the number of factors. *Multivariate Behav. Res.*, **1**, 245–276, 1966.

[24] Charikar M., and Sahai, A. Dimension reduction in L_1 norm. In *Proceedings of the 43rd Annual IEEE Symp. on Foundations of Computer Science*, 551–560, 2002.

[25] Coe, B., and Antler, C. Spot your genes - an overview of the microarray. *www.scq.ubc.ca/?p=272*, 2006.

[26] Cox, D. R. Regression Models and life tables (with discussion). *Statistical Society Series*, **B34**, 187, 1972.

[27] Dai, J. J., Lieu, L., and Rocke, D. M. Dimension reduction for classification with gene expression microarray data. *Statistical Applications in Genetics and Molecular Biology, vol. 5, issue 1, article 6*, 2006.

[28] Dasgupta, S. and Gupta, A. An elementary proof of the Johson- Lindenstrauss lemma. *Technical report 99-006*, UC Berkeley, March 1999.

[29] Dasgupta, S. Experiments with random projection. In *Uncertainty in Artificial Intelligence*, 2000.

[30] Datta, S., Le Rademacher, J., and Datta, S. Predicting patient survival by accelerated failure time modeling using partial least squares and lasso. *Biometrics*, **63**, 259–271, 2007.

[31] De Jong, S. SIMPLS: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, **18**, 251–263, 1993.

[32] Deegalla, S., and Bostrum, H. Reducing high-dimensional data by principal component analysis vs. random projection for nearest neighbor classification. In *Proc. of the 5th International Conference on Machine Learning and Applications*, 245–250, 2006.

[33] Denham, M. C. Implementing partial least squares. *Statistics and Computing*, **5**, 191–202, 1995.

[34] Dudoit, S., Gentleman, R., Irizarry, R., and Yang, Y.H. DNA microarray data oligonucleotide arrays. *Bioconductor short course*, 2003.

[35] Efron, B., Hastie, T. Johnstone, I. and Tibshirani, R. Least Angle Regression *Annals of Statistics*, **32**, 407–499, 2004.

[36] Engler, D. A., Li, Y. Survival analysis with large dimensional covariates: an application in microarray studies. *Harvard University Biostatistics Working Paper Series*, **68**, 2007.

[37] Fern, X. Z. and Brodley, C. E. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proc. of the Twentieth International Conference on Machine Learning*, 2003.

[38] Fradkin, D., and Madigan, D. Experiments with Random Projections for Machine Learning. *ACM*, 2002.

[39] Frankl, P. and Maehara, H. The Johnson-Lindenstrauss lemma and the sphericity of some graphs. *J. Combin. Theory Ser.*, **B44(3)**, 355–362, 1988.

[40] Geladi, P. Wold, Herman: The father of PLS. *Chemometrics and Intelligent Laboratory Systems*, **15.1**, R7–R8, 1992.

[41] Goel, N., Bebis G., and Nefian A. Face recognition experiments with random projection. *Proc. SPIE*, **5779**, 426–437, doi:10.1117/12.605553, 2005.

[42] Gui, J., and Li, H. Partial Cox regression analysis for high dimensional microarray gene expression data. *Bioinformatics*, **20**, 208–215, 2004.

[43] Gui, J., and Li, H. Penalized Cox regression analysis in the high-dimensional and low-sample size settings, with applications to microarray gene expression data. *Bioinformatics*, **21**, 3001–3008, 2005.

[44] Gui, J., and Li, H. Threshold gradient descent for censored data regression, with applications in pharmacogenomics. *Pacific Symposium on Biocomputing*, **10**, 272–283, 2005.

[45] Harr, B., and Schlotterer, C. Comparison of algorithms for the analysis of Affymetrix microarray data as evaluated by co-expression of genes in known operons. *Nucleic acid research*, **34(2): 8**, 2006.

[46] Hastie, T., Tibshirani, R. and Friedman, J. The elements of statistical learning. Data mining, inference, and prediction. *Springer*, New York, 2001.

[47] Hecht-Nielsen, R. Context vectors: General purpose approximate meaning representations self-organized from raw data. In *Computational Intelligence: Imitating Life (Zurada et al. eds.)*, 43–56, 1994.

[48] Hoskuldsson, A. PLS regression methods. *Journal of Chemometrics*, **2**, 211–228, 1988.

[49] Huang, J. and Harrington, D. Iterative Partial Least Squares with right-censored data analysis: a comparison to other dimension reduction techniques. *Biometrics*,

61, 17–24, 2005.

[50] Indyk, P., and Motwani, R. Appropriate nearest neighbors: towards removing the curse of dimensionality. In *Proc. 30th ACM Symp. on Theory of Computing*, 604–613, 1998.

[51] Indyk, P. Algorithmic applications in low-distortion embeddings. In *Proc. 42nd IEEE Symp Foundations of Computer Science*, 10–35, 2001.

[52] Jin Z, Lin D. Y., Wei L. J., Ying Z. L., Rank-based inference for the accelerated failure time model, *Biometrika*, **90**, 341–353, 2003.

[53] Joliffe, J. T. Principal component analysis. *Springer*, New York, 1986.

[54] Johnson, W. and Lindenstrauss, J. Extensions of Lipschitz maps into a Hilbert space. *Contemp. Math.*, **26**, 189–206, 1984.

[55] Kaiser, H. F. The varimax criterion for analytic rotation in factor analysis, *Psychometrika*, **23**, 187–200, 1958.

[56] Kaplan E. L., and Meier, P. Nonparametric estimation from incomplete observations. *Journal of American Statistics Association*, **53**, 467–481, 1958.

[57] Kaski, S. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proc. of IJCNN*, **26**, 413–418, Piscataway, NJ, 1998.

[58] Kharal, R. Semidefinite embedding for the dimensionality reduction of DNA microarray data. *Master Thesis in Computer Science, University of Waterloo*, 2006.

[59] Klein, J. P., and Moeschberger, M. L. Survival Analysis: techniques for censored and truncated data. *Springer*, second edition. New York, 2003.

[60] Kleinberg, J. M. Two algorithms for nearest-neighbor search in higher dimensions. In *Proc. of 29th ACM Symp. on Theory of Computing*, 599–608, 1997.

[61] Kohonen, T. et al. Self organization of massive document collection.

[62] Kurimo, M. Indexing audio documents by using latent semantic analysis and SOM. E. Oja and S. Kaski (eds.), Kohonen Maps, 1999.

[63] Lee, J. R. and Naor, A. Embedding the diamond graph in L_p and dimension reduction in L_1. *Geom Funct. Anal 14*, 745–747, 2004.

[64] Leung, K. M., Elashoff, R. M., and Afifi, A. A. Censoring issues in survival analysis. *Annual Review of Public Health*, **18**, 83–104, 1997.

[65] Leurgens, S. Linear models, random censoring and synthetic data. *Biometrika*, **74**, 301–309, 1987.

[66] Li, H. and Luan Y. Kernel Cox regression models for linking gene expression profiles to censored survival data. *Pacific Symposium of Biocomputing*, **8**, 65–76, 2003.

[67] Li, K. C. Sliced inverse regression for dimension reduction. *Journal of American Statistical Association*, **86**, 316–327, 1991.

[68] Li, K. C., Wang, J. L., and Chen C. H. Dimension reduction for censored regression data. *The Annals of Statistics*, **27**, 1–23, 1999.

[69] Li, L. and Li, H. Dimension reduction methods for microarrays with application to censored survival data. *Center for Bioinformatics and Molecular Biostatistics*, **Paper surv2**, 2004.

[70] Li, P., Hastie, T. J., and Church K. W. Very Sparse Random Projections. *Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining*, 287–296, 2006.

[71] Li, W., Bebis, G., and Bourbakis, N. Integrating algebraic functions of views with indexing and learnings for object recognition. In *IEEE Workshop on Learning in Computer Vision and Pattern Recognition*, 2004.

[72] Mardia, K. V., Kent, J. T., and Bibby, J. M. *Multivariate Analysis.* Academic Press, 2003.

[73] Martens, H., and Naes, T. *Multivariate calibration.* New York: Wiley, 1989.

[74] Marx, B. D. Iteratively reweighted partial least squares estimation for generalized linear regression. *Technometrics*, **38**, 374–381, 1996.

[75] Matousek, J. On variants of the Johnson-Lindenstrauss Lemma. *Wiley InterScience*, doi: 10.1002/rsa.20218, 2007.

[76] Ministry of Economic Development of New Zealand. `www.med.govt.nz/templates/MultipageDocumentPage____1065.aspx#`, 2005.

[77] National Center for Biotechnology Information. `http://www.ncbi.nlm.nih.gov/About/primer/microarrays.html`. Microarrays: chipping away at the mysteries of science and medicine. 2007.

[78] Nguyen, D. V. and Rocke, D. M. Partial least squares proportional hazard regression for application to DNA microarray survival data. *Bioinformatics*, **18**, 1625, 2002.

[79] Nguyen, D. V. and Rocke, D. M. On partial least squares dimension reduction for microarray-based classification: a simulation study. *Computational Statistics and Data Analysis*, **46**, 407–425, 2004.

[80] Nguyen, D. V. Partial least squares dimension reduction for microarray gene expression data with a censored response. *Mathematical Biosciences*, **193**, 119–137, 2005.

[81] Nguyen, T. N. and Rojo, J. Dimension reduction of microarray data in the presence of a censored survival response: a simulation study. *Statistical Applications in Genetics and Molecular Biology*, **8.1**, article 4. 2009.

[82] Nguyen, T. N. and Rojo, J. Dimension reduction of microarray gene expression data under the Accelerated Failure Time model. *Journal of Bioinformatics and Computational Biology*. In Press. To appear December 2009.

[83] Papadimitriou, C. H., Raghvan, P., Tamaki, H., and Vempala, S. Latent semantic analysis: A probabilistic analysis. In *Proc. of 17th ACM Symp. On the principles of Database Systems*, 159–168, 1998.

[84] Park, P. J., Tian, L., and Kohane I. S. Linking gene expression data with patient survival times using partial least squares. *Bioinformatics*, **20**, 208–215, 2002.

[85] Petricoin, E. F., Hackett, J. L., Lesko, L. J., Puri, R. K., Gutman, S. I., Chumakov, K., Woodcock, J., Feigal, D. W., Zoon, K. C., and Sistare, F. D. Medical applications of microarray technologies: a regulatory science perspective. *Nature genetics supplement*, **32**, 2002.

[86] Piatetsky-Shapiro, G., and Tamayo, P. Microarray data mining: facing the challenges. *SIGKDD Explorations*, **5.2**, 2004.

[87] Pinelis, I. On inequalities for sums of bounded random variables. Journal of Mathematical Inequalities **2.1**, 1–7, 2008.

[88] Ritov, Y., Estimation in a linear model with censored data, *Annals of Statistics* **18**, 303–328, 1990.

[89] Romanazzi, M., Influence in Canonical Correlation Analysis, *Biometrika* **57**, 237–259, 1992.

[90] Rosenwald, A., Wright, G., Chan, W. C., Connors, J. M., Campo E., Fisher R. I., Gascoyne R. D., Muller-Hermelink, H. K., Smeland, E.B., Giltnane, J. M., Hurt, E. M., Zhao, H., Averett, L., Yang, L., Wilson, W. H., Jaffe, E. S., Simon, R., Klausner, R. D., Powell, J., Duffey, P. L., Longo, D. L., Greiner, T. C., Weisenburger, D. D., Sanger, W. G., Dave, B. J., Lynch, J. C., Vose, J., Armitage, J. O., Monserrat, E., López-Gullermo, A., Grogran, T. M., Miller, T. P., LeBlanc, M., Ott, G., Kvaloy, S., Delabie, J., Holte, H., Krajci, P., Stokke, T., Staudt, L. M. The use of molecular profiling to predict survival after chemotherapy for diffuse large B-cell lymphoma. *New England Journal of Medicine*, **346**, 1937–1947, 2002.

[91] Siganov, I. S. Refinement of the upper bound of the constant in the central limit theorem. *Journal of Soviat Mathematics*, 2545–2550, 1986.

[92] Shi, L. DNA microarray (genome chip). `www.Gene-Chips.com`, 2002.

[93] Smyth G. K., and Speed, T. Normalization of cDNA microarray data. *Methods*, **31**, 265–273, 2003.

[94] Sun, J. Correlation principal component regression analysis of NIR data. *Journal of Chemometrics*, **9**, 21–29, 1995.

[95] Van Wieringen, W. N., Kun, D., Hampel, R. and Boulesteix, A. Survival prediction using gene expression data: a review and comparison. `http://www.slcmsr.net/boulesteix/papers/survival.pdf`.

[96] West, M., Blanchette, C., Dressman, H., Huang, E., Ishida, S., Spang, R., Zuzan, H., Olson, J. A., Marks, J. R., Nevins, J. R. Predicting the clinical status of human breast cancer by using gene expression profiles, *PNAS* **98.20**, 11462–11467, 2001.

[97] Whitehead, J. Fitting Cox's regression model to survival data using GLIM. *Applied statistics*, **29**, 268–275, 1980.

[98] Wikipedia. `http://en.wikipedia.org`.

[99] Wold, H. Estimation of principal components and related models by iterative least squares. In Krishnaiah, P. (ed.), *Multivariate Analysis*. Academic Press, N.Y., 391–420, 1966.

[100] Zhao, Q., and Sun, J. Cox survival analysis of microarray gene expression data using correlation principal component regression. *Statistical Applications in Genetics and Molecular Biology* **6.1**, article 16. Berkeley Electronic Press, 2007.

Appendix A

Appendix: Important Proofs, Algorithms

A.1 Dimension Reduction Methods

A.1.1 Principal Component Analysis (PCA)

PCA can be derived as an eigenvalue problem. The following theorem and corollaries from Mardia et al. [72] are essential in the derivation.

Theorem A.1 *Let A and B be two symmetric matrices. Suppose $B > 0$. Then,*

$$\max_x(\min_x)\ x^TAx \ s.t. \ \ x^TBx = 1$$

is attained when x is the eigenvector of $B^{-1}A$ corresponding to the largest (smallest) eigenvalue of $B^{-1}A$.

In other words, let λ_1 and λ_p be the largest and smallest eigenvalues of $B^{-1}A$, respectively, then $\lambda_1 = \max_x(x^TAx)$ and $\lambda_p = \min_x(x^TAx)$ subject to $x^TBx = 1$.

The following two corollaries from Mardia et al. [72] are also useful. So, we state them below.

Corollary A.1

If $R(x) = \frac{x^TAx}{x^TBx}$, then for $x \neq 0$,

$$\lambda_p \leq R(x) \leq \lambda_1$$

Corollary A.2

The maximum of a^Tx s.t. $x^TBx = 1$ is

$$(a^TB^{-1}a)^{1/2}$$

Further, $\max_x \frac{(a^T x)^2}{x^T B x} = a^T B^{-1} a$, *where the maximum is attained at* $x = \frac{B^{-1}a}{(a^T B^{-1} a)^{1/2}}$.

Returning to PCA and without loss of generality, suppose that the data matrix N x p data matrix X is centered (each column of X has mean 0), then the first weight vector can be obtained from the following optimization problem,

$$\begin{aligned} w_1 &= \arg\max_{w^T w=1} \mathrm{Var}(Xw) \\ &= \arg\max_{w^T w=1} (N-1)^{-1} w^T X^T X w \\ &= \arg\max \frac{w^T X^T X w}{w^T w}. \end{aligned}$$

Using theorem A.1 with $A = X^T X$ and $B = I$, we have w_1 as the eigenvector of $X^T X$ corresponding to the largest eigenvalue of $X^T X$. Thus, the first principal component (PC) is just $\tilde{x}_1 = X w_1$.

The second weight vector can be obtained from the following optimization problem,

$$\begin{aligned} w_2 &= \arg\max_{w^T w=1} \mathrm{Var}(Xw) \\ &= \arg\max_{w^T w=1} (N-1)^{-1} w^T X^T X w \end{aligned} \tag{A.1.1}$$

subject to $w_2^T X^T X w_1 = 0$. The last constraint ensures that the second PC, $\tilde{x}_2 = X w_2$, is orthogonal to the first PC, $\tilde{x}_1 = X w_1$. Now, let $P_1 = I - (\tilde{x}_1^T \tilde{x}_1)^{-1} \tilde{x}_1 \tilde{x}_1^T$, then P_1 is an N x N symmetric idempotent matrix ($P_1^2 = P_1$), which projects onto the subspace of $\mathbf{R}^n$ orthogonal to $\tilde{x}_1$. In other words, $P_1 \tilde{x}_1 = 0$. Hence, $P_1 X$ can be thought of as removing from the data matrix X the component that lies in the direction of the

first principal component. So, Eq. (A.1.1) can be written as

$$\begin{aligned} w_2 &= \arg\max_{w^T w=1} \mathrm{Var}(P_1 X w) \\ &= \arg\max_{w^T w=1} (N-1)^{-1} w^T X^T P_1 X w \\ &= \arg\max \frac{(N-1)^{-1} w^T X^T P_1 X w}{w^T w} \end{aligned}$$

which yields w_2 as eigenvector of $X^T P_1 X$ corresponding to the largest eigenvalue of $X^T P_1 X$. The second PC is $\tilde{x}_2 = X w_2$.

In general, let $P_{k-1} = I - \sum_{i=1}^{k-1} (\tilde{x}_i^T \tilde{x}_i)^{-1} \tilde{x}_i \tilde{x}_i^T$, then P_{k-1} is the projection matrix onto the subspace of R^n orthogonal to $\tilde{x}_1, \ldots, \tilde{x}_{k-1}$. The k^{th} weight vector can be obtained from the following optimization problem,

$$\begin{aligned} w_k &= \arg\max_{w^T w=1} \mathrm{Var}(X w) \\ &= \arg\max_{w^T w=1} (N-1)^{-1} w^T X^T X w \end{aligned} \tag{A.1.2}$$

subject to $w_k^T X^T X w_j = 0$, where $j = 1, \ldots, k-1$. Eq. (A.1.2) can be written as

$$\begin{aligned} w_k &= \arg\max_{w^T w=1} \mathrm{Var}(P_{k-1} X w) \\ &= \arg\max_{w^T w=1} (N-1)^{-1} w^T X^T P_{k-1} X w \\ &= \arg\max \frac{(N-1)^{-1} w^T X^T P_{k-1} X w}{w^T w} \end{aligned}$$

which yields w_k as eigenvalue $X^T P_{k-1} X$ corresponding to the largest eigenvalue of $X^T P_{k-1} X$. The k^{th} PC is $\tilde{x}_k = X w_k$.

Another approach to derive PCA is by using Lagrange multiplier in the optimization criteria. For example, the first weight vector can be written as,

$$w_1 = \arg\max_{w^T w=1} \mathrm{Var}(X w) = \arg\max_{w^T w=1} (N-1)^{-1} w^T X^T X w \tag{A.1.3}$$

The Lagrangian of Eq. (A.1.3) is

$$L(w, \lambda) = w^T X^T X w + \lambda(1 - w^T w).$$

Taking derivative with respect to w yields

$$\frac{\partial L}{\partial w} = 2X^T X w - 2\lambda w,$$

and setting to 0 gives

$$X^T X w = \lambda w$$

which yields w as the eigenvector of $X^T X$ corresponding to the largest eigenvalue of $X^T X$ (since the objective is to maximize the variance). The other weight vectors can be obtained similarly but with additional constraint $w_k^T X^T X w_j = 0$, where $j = 1, \ldots, k-1$.

A.1.2 Partial Least Squares (PLS)

PLS can be derived as an eigenvalue problem. The first weight vector can be obtained from the following optimization problem,

$$\begin{aligned} w_1 &= \arg\max_{w^T w = 1} \mathrm{Cov}(Xw, y) \\ &= \arg\max_{w^T w = 1} (N-1)^{-1} w^T X^T y \end{aligned} \tag{A.1.4}$$

where X and y are centered. The Lagrangian is given by

$$L(w, \lambda_1) = w^T X^T y + \lambda_1(1 - w^T w)$$

Taking derivative with respect to w yields

$$\frac{\partial L}{\partial w} = X^T y - 2\lambda_1 w,$$

and setting equal to 0 gives

$$X^T y = \lambda w \tag{A.1.5}$$

where we let $\lambda = 2\lambda_1$. Multiply (A.1.5) by w^T and enforcing the constraint $w^T w = 1$ give

$$w^T X^T y = \lambda w^T w = \lambda. \tag{A.1.6}$$

Taking the transpose of (A.1.6) yields

$$y^T X w = \lambda \tag{A.1.7}$$

Coupling (A.1.5) and (A.1.7) give the result,

$$\begin{pmatrix} 0 & X^T y \\ y^T X & 0 \end{pmatrix} \begin{pmatrix} w \\ 1 \end{pmatrix} = \begin{pmatrix} w \\ 1 \end{pmatrix}$$

which is just an eigenvalue problem of the form

$$A\tilde{w} = \lambda \tilde{w}$$

where $A = \begin{pmatrix} 0 & X^T y \\ y^T X & 0 \end{pmatrix}$, and $\tilde{w} = \begin{pmatrix} w \\ 1 \end{pmatrix}$.

The second weight vector can be obtained from the optimization problem,

$$\begin{aligned} w_2 &= \arg\max_{w^T w = 1} \mathrm{Cov}(Xw, y) \\ &= \arg\max_{w^T w = 1} (N-1)^{-1} w^T X^T y \end{aligned} \tag{A.1.8}$$

s.t. $w_2^T X^T X w_1 = 0$. The constraint ensures that the second PLS component, $\tilde{x}_2 = Xw_2$ is orthogonal to the first PLS component $\tilde{x}_1 = Xw_1$. Similar to PCA, we need to deflate the matrix X since we have to remove from the data matrix X the component

that lies in the direction of the first PLS component. Let $P_1 = I - (\tilde{x}_1^T \tilde{x}_1)^{-1} \tilde{x}_1 \tilde{x}_1^T$, then P_1 is an N x N symmetric idempotent matrix ($P_1^2 = P_1$), which projects onto the subspace of $\mathbf{R}^n$ orthogonal to $\tilde{x}_1$. In other words, $P_1 \tilde{x}_1 = 0$. So, Eq. (A.1.8) can be written as

$$\begin{aligned} w_2 &= \arg\max_{w^T w=1} \mathrm{Cov}(P_1 X w, y) \\ &= \arg\max_{w^T w=1} (N-1)^{-1} w^T X^T P_1 y \end{aligned} \tag{A.1.9}$$

which results in the eigenvalue problem $A\tilde{w} = \lambda \tilde{w}$, where $A = \begin{pmatrix} 0 & X^T P_1 y \\ y^T P_1 X & 0 \end{pmatrix}$, and $\tilde{w} = \begin{pmatrix} w \\ 1 \end{pmatrix}$. The second PLS component is just $\tilde{x}_2 = X w_2$.

In general, let $P_{k-1} = I - \sum_{i=1}^{k-1} (\tilde{x}_i^T \tilde{x}_i)^{-1} \tilde{x}_i \tilde{x}_i^T$, then P_{k-1} is the projection matrix onto the subspace of R^n orthogonal to $\tilde{x}_1, \ldots, \tilde{x}_{k-1}$. The k^{th} weight vector can be obtained from the following optimization problem,

$$\begin{aligned} w_k &= \arg\max_{w^T w=1} \mathrm{Cov}(Xw, y) \\ &= \arg\max_{w^T w=1} (N-1)^{-1} w^T X^T y \end{aligned}$$

subject to $w_k^T X^T X w_j = 0$, where $j = 1, \ldots, k-1$. This is equivalent to

$$\begin{aligned} w_k &= \arg\max_{w^T w=1} \mathrm{Cov}(P_{k-1} X w, y) \\ &= \arg\max_{w^T w=1} (N-1)^{-1} w^T X^T P_{k-1} y \\ &= \arg\max \frac{(N-1)^{-1} w' X' P_{k-1} y}{w^T w=1} \end{aligned} \tag{A.1.10}$$

which yields w_k solution to the eigenvalue problem $A\tilde{w} = \lambda \tilde{w}$, where

$$A = \begin{pmatrix} 0 & X^T P_{k-1} y \\ y^T P_{k-1} X & 0 \end{pmatrix}, \text{ and } \tilde{w} = \begin{pmatrix} w \\ 1 \end{pmatrix}.$$

The k^{th} PLS component is just $\tilde{x}_k = X w_k$.

Another approach to derive the PLS weights is as follows. The first weight vector is obtained from the optimization criteria of (A.1.4), which can be rewritten as

$$\begin{aligned} w_1 &= \arg\max (N-1)^{-1} \frac{w^T X^T y}{w^T w} \\ &= \arg\max (N-1)^{-1} \frac{y^T X w}{w^T w} \end{aligned}$$

From corollary A.1.2, with $a = X^T y$, and $B = I$, we obtain $w_1 = \frac{X^T y}{||X^T y||}$. The first PLS component is $\tilde{x}_1 = X w_1$. Similarly, the second weight vector is obtained from the optimization criteria of (A.1.9), which is equivalent to

$$\begin{aligned} w_2 &= \arg\max (N-1)^{-1} \frac{w^T X^T P_1 y}{w^T w} \\ &= \arg\max (N-1)^{-1} \frac{y^T P_1 X w}{w^T w} \end{aligned}$$

Using corollary A.1.2, we get $w_2 = \frac{X' P_1 y}{||X^T P_1 y||}$, and the second PLS component is $\tilde{x}_2 = X w_2$.

In general, the k^{th} weight vector is obtained from (A.1.10). Again, using corollary A.1.2, we get $w_k = \frac{X^T P_{k-1} y}{||X^T P_{k-1} y||}$, and the k^{th} PLS component is $\tilde{x}_k = X w_k$.

A.1.3 Sliced Inverse Regression (SIR) Algorithm

The double slicing algorithm of Li et al. [68] for a censored response is provided below.

For $i = 1, \ldots, N$, partition (T_i, δ_i) for the two cases: uncensored ($\delta_i = 1$), and censored ($\delta_i = 0$). Here, T_i denotes the minimum of (y_i, c_i), $i = 1, \ldots, N$, where y_i's

are the true survival times, and c_i's are the right-censoring times. Let N_0 = number of $\delta_i = 0$, N_1 = number of $\delta_i = 1$, and $N_0 + N_1 = N$.

1) Divide into equal quantiles the range of $(T_i, \delta_i = 1)$ into m_1 slices, $J_1, \ldots, J_{m_1}$. Divide into equal quantiles the range of $(T_i, \delta_i = 0)$ into m_0 slices, $I_1, \ldots, I_{m_0}$. Let $m = m_0 + m_1$.

2) Calculate sample mean: $u = \frac{1}{N}\sum_{i=1}^{N} X_i$. In other words, the p x 1 vector u is obtained by averaging out the rows of X.

For $\delta_i = 1$, calculate the sliced mean $u_{1d} = \frac{1}{n_{1d}}\sum_{y_i \in J_d} X_i$, where $d = 1, \ldots, m_1$, and n_{1d} denotes the cardinality of slice J_d. Similarly, for $\delta_i = 0$, calculate $u_{0c} = \frac{1}{n_c}\sum_{y_i \in I_c} X_i$, where $c = 1, \ldots, m_1$, and n_{0c} denotes the cardinality of slice I_c.

3) Calculate $\hat{\Sigma}_x = \frac{1}{N}\sum_{i=1}^{N}(X_i - u)(X_i - u)'$,

and

$\hat{\Sigma}_b = \frac{1}{N}\left(\sum_{c=1}^{m_0} n_{0c}(u_{0c} - u)(u_{0c} - u)' + \sum_{d=1}^{m_1} n_{1d}(u_{1d} - u)(u_{1d} - u)'\right)$.

4) Solve the eigenvalue decomposition: $v_i = \lambda_i \hat{\Sigma}_b^{-1}\hat{\Sigma}_x v_i$, where $\lambda_1 \geq \cdots \geq \lambda_p \geq 0$. The v_i is the i^{th} SIR weight vector.

A.2 Regression Methods for Right-Censored Survival Data

Censoring occurs when the available information regarding the survival of some individuals is incomplete (Leung et al. [64]). In practice, survival data are usually censored or incomplete. An observation is said to be *right-censored* if the subject is still alive at the termination of the study or is lost to follow-up at any time during the study such that the survival time of that subject y is only known to exceed a certain value, denoted by right censoring time c_r (Leung et al. [64]). An observation is said to be *left-censored* if the subject's survival time y is less than a left censoring time c_l. Here, the event of interest has already occurred for the subject before the

subject is observed in the study at time c_l (Klein and Moeschberger [59]). *Interval censoring* occurs when the survival time of the subject is only known to occur in an interval (Klein and Moeschberger [59]).

Another important feature of survival data is truncation. Truncation occurs when only those subjects whose survival time lies within an observational window, denoted by interval (y_l, y_r), are observed (Klein and Moeschberger [59]). The key difference between censoring and truncation is that with censoring, there is at least partial information on the subjects, while with truncation, no information is known for the subjects whose survival time lies outside the interval (y_l, y_r). *Left truncation* occurs when y_r is infinite, *right truncation* occurs when y_l is zero, and *interval truncation* occurs when $y_l > 0$ and $y_r < \infty$.

A.2.1 Kaplan-Meier and Nelson-Aalen Estimators of the Survival Function

A popular method to estimate the survival function taking into account right-censored information without incorporating the covariates is the Kaplan-Meier estimator, also known as the product-limit estimator (Kaplan and Meier [56]). Suppose for D distinct times, $t_1 < t_2 < \cdots < t_D$, and there are d_i events (deaths) and N_i subjects who are at risk at time t_i. The Kaplan-Meier estimator is defined as:

$$\hat{S}_{KM}(t) = \begin{cases} 1 & \text{if } t < t_1 \\ \prod_{t_i \leq t} \left[1 - \frac{d_i}{N_i}\right] & \text{if } t_1 \leq t \end{cases}$$

For values beyond the largest observation time t_{max}, the Kaplan-Meier estimator is not well-defined. This is because if t_{max} is a death time, then the estimated survival curve is zero beyond t_{max}, but if t_{max} is censored, then the survival curve is undetermined beyond t_{max}. Several solutions were described in Klein and Moeschberger [59]. As

noted in [59], the Kaplan-Meier estimator is a step function with jumps at the observed event times, and the size of these jumps depends on both the number of events observed at each t_i and the pattern of the censored observations prior to t_i. When there is no censoring, the Kaplan-Meier estimator reduces to the empirical survival function. Using Greenwood's formula, the estimated variance of the Kaplan-Meier estimator is given by:

$$\hat{V}[\hat{S}_{KM}(t)] = \left(\hat{S}_{KM}(t)\right)^2 \sum_{t_i \leq t} \frac{d_i}{N_i(N_i - d_i)}$$

and the standard error is $\sqrt{\hat{V}[\hat{S}_{KM}(t)]}$. Using the Kaplan-Meier estimator, we can estimate the cumulative hazard function $H(t)$ by $\hat{H}_{KM}(t) = -log[\hat{S}_{KM}(t)]$.

An alternative estimator of the cumulative hazard function $H(t)$ is the Nelson-Aalen estimator, defined as:

$$\hat{H}_{NA}(t) = \begin{cases} 0 & \text{if } t < t_1 \\ \sum_{t_i \leq t} \frac{d_i}{N_i} & \text{if } t_1 \leq t \end{cases}$$

and the estimated variance for the Nelson-Aalen estimator is given by:

$$\hat{V}[\hat{H}_{NA}](t) = \sum_{t_i \leq t} \frac{d_i}{N_i^2}$$

Using the Nelson-Aalen estimator for cumulative hazard function $H(t)$, we can estimate the survival function as $\hat{S}_{NA}(t) = e^{-\hat{H}_{NA}(t)}$. We should note that both the Kaplan-Meier and Nelson-Aalen estimators are based on the assumption that true survival time and censoring time are independent (non-informative censoring). In other words, knowledge of the subject's censoring time provides no further information about that subject's likelihood of survival at a future time if the subject continued on with the study (Klein and Moeschberger [59]).

A.2.2 Accelerated Failure Time (AFT) Model

We describe in details three parametric models for the AFT model: Exponential, Weibull, and Lognormal.

Exponential and Weibull distribution

We drop the subscript i (which denotes the i^{th} observation) in the model (4.2.4) for convenience. The linear model for the logarithm of the true survival time is

$$log(Y) = \mu + Z\beta + \sigma U \tag{A.2.1}$$

where U has an extreme value distribution. In other words,

$$f_U(u) = exp(u - e^u), \quad u \in (-\infty, \infty),$$

and

$$S_U(u) = exp(-e^u)$$

This implies

$$f_Y(y) = exp\left(\frac{(log(y) - Z^*)}{\sigma} - exp\left(\frac{(log(y) - Z^*)}{\sigma}\right)\right)$$

and

$$S_Y(y) = exp\left(-exp\left(\frac{(log(y) - Z^*)}{\sigma}\right)\right) \tag{A.2.2}$$

where $Z^* = \mu + Z\beta$. We should note that Eq. (A.2.2) can be written as

$$S_Y(y) = exp\left(-y^{1/\sigma}e^{-Z^*/\sigma}\right) \tag{A.2.3}$$

and hence, Y has a Weibull distribution with shape parameter $\alpha = 1/\sigma$, and scale parameter $\lambda = e^{-Z^*/\sigma}$. If $\sigma = 1$, then we Y has an exponential distribution with rate parameter $\lambda = e^{-Z^*}$.

Lognormal distribution

Assuming the log linear model in (A.2.1) with $U \sim N(0,1)$. This implies

$$log(Y) \sim N(Z^*, \sigma^2)$$

and

$$Y \sim LN(Z^*, \sigma^2).$$

Thus,

$$S_Y(y) = \frac{1}{2}\left[1 - erf\left(\frac{log(y) - Z^*}{\sigma\sqrt{2}}\right)\right]$$

where

$$erf(w) = \frac{2}{\sqrt{\pi}}\int_0^w e^{-t^2}dt.$$

A.3 Simulation Setup

The details to the simulation procedure for generating the gene expression values and the survival times are provided below.

A.3.1 Generating Gene Expression Values

Let x_{ij} be the ij^{th} entry of the gene expression data matrix X, where $i = 1, \ldots, N$ denote the indices for the subjects, and $j = 1, \ldots, p$ denote the indices for the gene. We generate $x^*_{ij} = \sum_{l=1}^{d} r_{li}\tau_{lj} + \epsilon_{ij}$, for $l = 1, \ldots, d$, where $\tau_{lj} \sim N(\mu_\tau, \sigma^2_\tau)$ are the component values, and $\epsilon_{ij} \sim N(\mu_\epsilon, \sigma^2_\epsilon)$ are the noise. The ij^{th} entry of the gene expression data matrix is $x_{ij} = exp(x^*_{ij})$. Thus, the gene expressions are generated as a linear combination of the d underlying components and an error component. It is clear that $x_{ij} \sim LN(a_i, b_i^2)$, with parameters $a_i = \mu_\tau \sum_{l=1}^{d} r_{li}$, and $b_i^2 = \sigma^2_\tau \sum_{l=1}^{d} r^2_{li} + \sigma^2_\epsilon$. As pointed out by Nguyen [81], the gene expression data matrix is generated so

that the first k principal components explain a specified proportion of variability in the data matrix, and the total variation explained (TVPE) by the first k principal components is controlled in the simulation by $\delta = \sigma_\epsilon/\sigma_\tau$. In this simulation setup, we fix $d = 6$, $\mu_\epsilon = 0$, $\mu_\tau = 5/d$, $\sigma_\tau = 1$. An important aspect of the simulation is to select k, the dimension of the reduced data matrix after applying dimension reduction methods. We consider 2 cases: 1) fix k across all methods by varying σ_ϵ so as to capture the desired TVPE for the first k PC's, namely 40%, 50%, 60% and 70%, and 2) select k by cross-validation, and set $\sigma_\epsilon = 0.3$. For each $p \in \{100, 300, 500, 800, 1000, 1200, 1400, 1600\}$, 5000 datasets are generated. Since we want to consider $p \gg N$, the sample size $N = 50$ is fixed. Since r is a set of fixed constants, it is convenient to select $r_{li} \sim Unif(-0.2, 0.2)$, and we use the same set of r for all the simulations (see Nguyen and Rojo [81] for discussion on the choice of r_{li}).

A.3.2 Generating Survival and Censoring Times

Once the gene expression data matrix X have been generated, the survival time of the i^{th} individual, y_i, is generated independently from the censoring time, c_i, with $(i = 1, \ldots, N)$. For the Cox model, the Proportional Hazards (PH) assumption needs to be satisfied. Thus, y_i takes the following form:

$$y_i = g(y_{0i}, x_i'\beta).$$

Here, y_{0i} denotes the baseline survival time for the i^{th} individual, g is a function of both the baseline survival and covariates, and is assumed to be a monotonically increasing transformation of y_{0i} that satisfies the PH assumption. To obtain a close form for the true censoring rate $P[y_i > c_i]$, the censoring times c_i's are generated similarly to the true survival times, i.e.,

$c_i = g(c_{0i}, x_i'\beta)$,

where c_{0i} denotes the baseline censoring time for the i^{th} individual. This work considers the Exponential and Weibull distributions for the simulations, which are described below.

For the Exponential baseline survival with density $f_0(t) = \lambda e^{-\lambda t}$ and survival $S_0(t) = e^{-\lambda t}$, i.e. $y_{0i} \sim Exp(\lambda_y)$, and $c_{0i} \sim Exp(\lambda_c)$, the true survival and censoring times are:

$y_i = y_{0i}e^{-x_i'\beta}$ and $c_i = c_{0i}e^{-x_i'\beta}$.

The observed survival time for the i^{th} individual is $T_i = min(y_i, c_i)$, and the corresponding censoring indicator is $\delta_i = I(y_i < c_i)$, with $\delta_i = 1$ for death event and $\delta_i = 0$ for the censored response. The true censoring rate, $P[y_i > c_i]$, is obtained as $P[y_i > c_i] = \frac{\lambda_c}{\lambda_y + \lambda_c}$. In the simulations, we fix $\lambda_y = 2$, and vary λ_c to obtain the desired amount of censoring of 1/3, and 1/2. For example, if $\lambda_c = 1$, then the censoring rate is 1/3.

For the Weibull baseline survival with density $f_0(t) = \left(\frac{a}{b}\right)\left(\frac{t}{b}\right)^{a-1} e^{-(t/b)^a}$, and $S_0(t) = e^{-(t/b)^a}$, i.e. $y_{0i} \sim Weibull(a, b_y)$, and $c_{0i} \sim Weibull(a, b_c)$, the true survival and censoring times are:

$y_i = y_{0i}(e^{-x'\beta})^{1/a}$ and $c_i = c_{0i}(e^{-x'\beta})^{1/a}$.

The true censoring rate, $P[y_i > c_i]$, is obtained as $P[y_i > c_i] = \frac{(b_y)^a}{(b_y)^a + (b_c)^a}$. In the simulation setup, we fix $a = 5$, and $b_y = 2^{1/a}$ and vary b_c to obtain the desired amount of censoring of 1/3, and 1/2. For example, if $b_c = 4^{1/a}$, then the censoring rate is 1/3. The results for the Weibull distribution are similar to those for the Exponential case.

For the AFT model, the survival time of the i^{th} individual, y_i, is generated independently from the censoring time, c_i, with $(i = 1, \ldots, N)$, as follows:

$\ln(y_i) = \mu + X_i'\beta + u_i$, and $\ln(c_i) = \mu + X_i'\beta + w_i$.

Here, X_i is the vector of covariates corresponding to the i^{th} individual. We set $\mu = 0$, and consider an exponential, lognormal, log-t, and lognormal mixture model for the true life times. For example, in the case of the exponential model, the errors u_i's are taken to be from a standard extreme value distribution, with density $f_{u_i}(t) = e^{t-e^t}$ for $-\infty < t < \infty$. The error for the censoring times w_i are taken to be from an exponential distribution, i.e. $w_i \sim Exp(\lambda_c)$, with density $f_{w_i}(t) = \lambda_c e^{\lambda_c t}$. The λ_c is varied in these simulations to obtain a censoring rate of $1/3$. The true censoring rate is $P[y_i > c_i] = P[u_i > w_i] = \int_0^\infty S_{u_i}(t) f_{w_i}(t) dt$. The observed survival time for the i^{th} individual is $T_i = min(y_i, c_i)$, and the corresponding censoring indicator is $\delta_i = I(y_i < c_i)$. In the case of the lognormal mixture model, the errors u_i are taken to be from a normal mixture distribution, with density $f_{u_i}(x) = 0.9\phi(x) + \frac{0.1}{10}\phi\left(\frac{x}{10}\right)$. In the case of the lognormal model, the errors u_i are taken to be from a standard normal distribution. In the case of the log-t model, the errors $u_i \sim t(3)$. The error for the censoring times w_i are taken to be from a Gamma distribution $Gamma(a_C, s_C)$, with $a_C = 3$, and s_C chosen such that the censoring rate is $1/3$ for the lognormal mixture, lognormal and log-t models.

A.3.3 Generating the Weights on the Genes

The goal of the simulation study is to assess the performance of the different dimension reduction methods in the presence of outliers. Since the true regression parameters, β_j with $j = 1, \ldots, p$, are fixed, it is convenient to generate them from a $N(0, \sigma_\pi^2)$ distribution. In these simulations, $\sigma_\pi = 0.2$ is fixed for all p's (the number of genes). By fixing $\sigma_\pi = 0.2$, the absolute values of $X_i'\beta$ are increased for large values of p. Since both the true and censoring times for the i^{th} individual depend on $X_i'\beta$, the observed survival times will have outliers for large values of p. Moreover, for the AFT

model, the survival times generated from the lognormal mixture model tend to have a longer tail than those generated from the exponential or lognormal model, and thus, outliers are more likely to be present in the response under the lognormal mixture model than the exponential or lognormal model.

A.3.4 Selection of k

Since the reduced data matrix is of dimension p x k after applying dimension reduction methods to the original data matrix, two scenarios for the selection of k are considered for the different dimension reduction methods under the Cox model: 1) k is fixed across the different methods, and 2) k is selected based on the minimization of the cross-validation squared error of the estimated survival function for each method. Since $p \gg N$ in microarray data, sample size of $N = 50$ is chosen, and the number of genes, $p = 100$, 300, 500, 800, 1000, 1200, 1400, and 1600, are considered. 5000 data sets are generated, and for each dataset, dimension reduction methods are applied in stage 1, and the data in the reduced subspace is used with the Cox PH model in stage 2. Several dimension reduction methods are considered: PCA, MPLS, RMPLS, SIR, UNIV, SPCR, and CPCR.

For scenario 1, we fix $k = 3$ for all the methods. Since the data matrix is generated so that the first k PCs explain a specified proportion of predictor variability, we set the proportion of variability explained to be 40%, 50%, 60% and 70%. We should note that for SIR, we first reduce the dimension of the data matrix from p to $k = 3$ via PCA or MPLS, then apply SIR to the reduced subspace and obtain $k_{SIR} = 2$ SIR components. For Univariate Selection (UNIV), we fit a univariate Cox model for each gene, then obtain $k = 3$ most important genes according to the rank of the p -values of the coefficient in the univariate Cox model. For Supervised Principal Component

Regression (SPCR), we first select $\lambda_{SPCR} = 20\%$ of the genes by UNIV, then apply PCA to the λ_{SPCR} genes to obtain the $k = 3$ SPCR components. For Correlation Principal Component Regression (CPCR), we first apply PCA to the original data matrix to obtain $\lambda_{CPCR} = min(N, p)$ PCs, then apply UNIV to the resulted PCs to obtain the $k = 3$ CPCR components. For scenario 2, we allow adaptive tuning for each method by use of cross-validation (CV). We exclude SIR from the analysis because the method does not improve on PCA or MPLS. Also, for SPCR, we fix $\lambda_{SPCR} = 20\%$, and apply cross-validation to select k.

Under the AFT model, we select k based on the minimization of the cross-validation squared error of fit or squared residuals of the log lifetimes for each method, and compare RMPLS, RRWPLS, RMIPLS to MPLS, RWPLS, MIPLS, PCA, UNIV, CPCR, and SCPR.

For the Random Projection (RP), we set $N = 50$, $p = 3000$ and $\epsilon = 0.15$, and consider the Cox model. We compare the two procedures: 1) using PCA or RMPLS, and 2) combining RP and PCA or RMPLS. For procedure 1, we use dimension reduction methods of PCA or RMPLS to the original data matrix X, and then apply the Cox model to the reduced data matrix. For procedure 2, we first apply RP to the original data matrix using a random projection matrix of dimension p x $k.r$, where $k.r$ is obtained from the various lower bounds for k considered in this work, then apply PCA and RMPLS, and finally use the Cox model with the reduced data matrix obtained by first using RP then using PCA or RMPLS.

A.4 R Code

A.4.1 Rank-based Modified Partial Least Squares (RMPLS)

The algorithm for RMPLS is based on the orthogonal scores algorithm (section 2.7) taking into account the censoring using Nguyen and Rocke's procedure of MPLS by replacing the dot product of U (eigenvectors of XX^T) and y (the response) with the slope coefficient obtained from the univariate Cox or AFT regression of y on U.

Instead of using covariate data matrix X and and the response y, RMPLS uses R_X and R_y, the ranks of the columns of X and the ranks of y, respectively.

```
###################Description#####################
###Rank-based Modified Partial Least Squares###
##The function nipals provide the code for RMPLS##


    ##Arguments##
#X: covariate matrix: N x p#
#Y: univariate response: N x 1#
#del: censoring indicator vector: N x 1:  del=0 means no censoring#
 #otherwise is a vector of censoring indicators#


#cor.method=1 denotes the Pearson correlation; =2 denotes spearman #
 #rank correlation used in objective criterion of PLS#


#method=1: cox model; =2: AFT exponential model; =3: AFT lognormal #
 #model (need the scale parameter); =4: other considered
 #distributions such as lognormal mixture, log-t (need to specify #
```

```
 #the distribution (my.dist) and the parameters (parms)) #

#-------------------------------------------------------------#
#-----function nipals for RMPLS-----#
#default arguments: del=0 (no censoring), cor.method=1 (Pearson
 #correlation), scale=1 (AFT lognormal model scale parameter), #
 #my.dist=0 (no distribution used for AFT), parms=0 (no #
 #parameters used)#

nipals<-function (X, Y, del=0, ncomp,cor.method=1,
method=1,scale=1,stripped = FALSE,my.dist=0,parms=0, ...)
{
    Y <- as.matrix(Y);Y.orig<-Y   #original Y#
    if (!stripped) {
        dnX <- dimnames(X)
        dnY <- dimnames(Y)
    }
    dimnames(X) <- dimnames(Y) <- NULL
    nobj <- dim(X)[1]            #number of observations, N#
    npred <- dim(X)[2] #number of covariates, p#
    nresp <- dim(Y)[2] #number of response vectors (taken =1)#

#Initialization#
# the weight matrix P, and temporary matrices V & R#
V <- R <- matrix(0, nrow = npred, ncol = ncomp)
```

```
    tQ <- matrix(0, nrow = ncomp, ncol = nresp)
    B <- array(0, dim = c(npred, nresp, ncomp))
    if (!stripped) {
        P <- R
        U <- TT <- matrix(0, nrow = nobj, ncol = ncomp)
        fitted <- array(0, dim = c(nobj, nresp, ncomp))
    }

    #---------standardize X & Y (mean 0, variance 1)---------#
        Xmeans <- colMeans(X)
        X.sd  <- apply(X,2,sd)
         ##X: each column has mean 0, variance 1##
        X <- (X - rep(Xmeans, each = nobj))/rep(X.sd,each=nobj)
        Ymeans <- colMeans(Y)
        Y.sd<-apply(Y,2,sd)
         ##Y: vector Y has mean 0, variance 1##
Y <- (Y - rep(Ymeans, each = nobj))/rep(Y.sd,each=nobj)
    #-----------------------------------------------------#
    #--------Compute the S=Cor(X,Y): ---------------------#
     #cor.method=1: Pearson correlation#
     #cor.method=2: Spearman correlation#
    if(cor.method==1) {S<-crossprod(X,Y)}
    if(cor.method==2) {S<-cor(X,Y,method="spearman")}
    #-----------------------------------------------------#
```

```
    ##-----find q.a, the eigenvectors of S---##
    # if y is univariate, then q.a=1 (since eigenvector #
 # for a vector is 1 #
# if y is multivariate (more than 1 column), then #
 # q.a is the leading standardized eigenvector of S#
for (a in 1:ncomp) {
        if (nresp == 1) {
            q.a <- 1
        }
        else {
            if (nresp < npred) {
                q.a <- eigen(crossprod(S), symmetric = TRUE)
$vectors[, 1]
            }
            else {
                q.a <- c(crossprod(S, eigen(S %*% t(S),
symmetric = TRUE)
$vectors[, 1]))
                q.a <- q.a/sqrt(c(crossprod(q.a)))
            }
        }

        #---Find r.a, the crossproduct of S & q.a------#
#---and t.a, the crossproduct of X and r.a--------#
        r.a <- S %*% q.a
```

```
        t.a <- X %*% r.a

        #-------Finding the weight vectors p.a according to the --#
 #algorithm in section 4.1-------------#
         #in the case for Pearson correlation: only consider the#
          #crossproduct of the covariate data matrix X#
          #and the response y#
if(cor.method==1){
          tmp <- t.a-mean(t.a); tnorm<-sqrt(c(crossprod(tmp)))
          p.a<-(crossprod(X,X)
%*%S%*%q.a - t(X)%*%as.matrix(rep(mean(t.a),nobj),ncol=1))/tnorm
        }

 #in the case for Spearman rank correlation: use the #
          #crossproduct of R_X and R_y instead of X and y#
          # note the function cov(A,B,method="spearman") will #
          # compute the Spearman rank covariance for each #
          # column of A with each column of B#
if(cor.method==2){
          tmp <- t.a-mean(t.a); tnorm<-sqrt(c(crossprod(tmp)))
          p.a.tmp<-(cov(X,X,method="spearman")
%*%S%*%q.a*(nobj-1) - t(X)%*%as.matrix(rep(mean(t.a),
nobj), ncol=1))/tnorm
   #the weight vector p.a is standardized#
          p.a<-p.a.tmp/as.numeric(sqrt(crossprod(p.a.tmp,p.a.tmp)))
```

```
        }

      #readjust and standardize t.a, and r.a (mean 0, variance 1)#
      # these are temporary vectors using to compute the weight#
      # vector p.a#
        t.a <- t.a - mean(t.a)
        tnorm <- sqrt(c(crossprod(t.a)))
        t.a <- t.a/tnorm
        r.a <- r.a/tnorm

        #-----------Using MPLS to incorporate censoring: --#
 #------modify q.a to incorporate censoring---#
 #no censoring: q.a is slope coef of univariate#
  #regression of y on t.a#
             #no censoring#
  if(length(del)==1) {
              q.a <- crossprod(Y, t.a)}

 #censoring: q.a is slope coef from Cox or AFT #
#regression of y on t.a#
if(length(del) > 1){
              #method=1: Cox regression#
              if(method==1){
                q.a <- coxph(Surv(Y,del)~t.a)$coef[1]/tnorm
              }
```

```
   #method=2: AFT exponential model#
               if(method==2){
                 q.a <- survreg(Surv(Y.orig,del)~t.a,
dist="exponential")$coef[[2]]/tnorm
               }


                #method=3: AFT lognormal model#
               if(method==3){
                 q.a <- survreg(Surv(Y.orig,del)~t.a,
dist="lognormal")$coef[[2]]/tnorm
               }


   #method=4: other models for AFT: include log-t, #
#lognormal mixture#
# the distribution is specified by my.dist with#
# parameters in parms#
               if(method==4){
                 q.a <- survreg(Surv(Y.orig,del)~t.a,dist=my.dist,
parms=parms)$coef[[2]]/tnorm
               }
            }


#Calculate v.a=p.a (a temporary vector used to adjust S) #
          #and standardize it#
```

```
        v.a <- p.a
        if (a > 1) {
            v.a <- v.a - V %*% crossprod(V, p.a)
        }
        v.a <- v.a/sqrt(c(crossprod(v.a)))

        #deflate S=Cor(X,Y), for each subsequent#
 #weight vector according to the algorithm#
        S <- S - v.a %*% crossprod(v.a, S)

#put everything in a matrix#
#P: weight matrix with columns as weight vectors#
R[, a] <- r.a
        tQ[a, ] <- q.a
        V[, a] <- v.a
        B[, , a] <- R[, 1:a, drop = FALSE] %*% tQ[1:a, ,
 drop = FALSE]
        if (!stripped) {
            u.a <- Y %*% q.a
            if (a > 1)
                u.a <- u.a - TT %*% crossprod(TT, u.a)
            P[, a] <- p.a
            TT[, a] <- t.a
            U[, a] <- u.a
            fitted[, , a] <- TT[, 1:a] %*% tQ[1:a, ,
```

```
drop = FALSE]
        }
    }
    if (stripped) {
        list(coefficients = B, Xmeans = Xmeans, Ymeans = Ymeans)
    }

    #Output#
    else {
        residuals <- -fitted + c(Y)
        fitted <- fitted + rep(Ymeans, each = nobj)
        objnames <- dnX[[1]]
        if (is.null(objnames))
            objnames <- dnY[[1]]
        prednames <- dnX[[2]]
        respnames <- dnY[[2]]
        compnames <- paste("Comp", 1:ncomp)
        nCompnames <- paste(1:ncomp, "comps")
        dimnames(TT) <- dimnames(U) <- list(objnames, compnames)
        dimnames(R) <- dimnames(P) <- list(prednames, compnames)
        dimnames(tQ) <- list(compnames, respnames)
        dimnames(B) <- list(prednames, respnames, nCompnames)
        dimnames(fitted) <- dimnames(residuals) <- list(objnames,
            respnames, nCompnames)
        class(TT) <- class(U) <- "scores"
```

```
        class(P) <- class(tQ) <- "loadings"

        ##########################################
        #Of interest is the weight matrix for RMPLS#
        #weight vectors: loadings, components: scores#
        list(coefficients = B, scores = TT, loadings = P,
 Yscores = U,
            Yloadings = t(tQ), projection = R, Xmeans = Xmeans,
            Ymeans = Ymeans, fitted.values = fitted,
 residuals = residuals,
            Xvar = colSums(P * P), Xtotvar = sum(X * X))
    }
}
```

A.4.2 Accelerated Failure Time (AFT) Model: Implement the Log-normal Mixture Model

The following code is added to the library *survival* in R with modifications to the function *survreg* to incorporate the log-normal mixture distribution for the Accelerated Failure Time model. The modifications require an object for the distribution of log-normal mixture, a function to calculate the mean and variance of the distribution, a function to calculate the log-likelihood, a function to calculate the density, and a function to calculate the quantiles.

```
#call library survival#
library(survival)
 #Error density for lognormal mixture model: #
```

```
  #phi(.) denotes the pdf of standard normal (Gaussian)#
  #w's are the weights for the mixture#
  #f_e(x) = w1*phi(x,mu,sd1)+w2*phi(x,mu,sd2),#
  #w1+w2=1#

 #define normal mixture (norMix) dist. in AFT model #
  #survreg package: attributes in survreg.distributions#
   #name: norMix (normal mixture)#
  survreg.distributions$norMix$name<-"norMix"
  #variance for norMix: parms consist of vector of weights,#
   #mean, standard deviation #
  survreg.distributions$norMix$variance<-function (parms){
  w<-parms[1:2];mu<-parms[3:4];sigma<-parms[5:6]
   #make a norMix object consisting of the means, #
    #standard deviations, and weights #
  obj<-norMix(mu=mu,sig2 = sigma^2,w=w)
    #return the variance of the object#
  var.norMix(obj)
}

 #initialize: mean and variance for the norMix object#
survreg.distributions$norMix$init<-function (x, weights,parms)
{
    w<-parms[1:2];mu<-parms[3:4];sigma<-parms[5:6]
     obj<-norMix(mu=mu,sig2 = sigma^2,w=w)
```

```
    if (sigma[1] <= 0 | sigma[2] <= 0)
        stop("Invalid sd for the normal distribution")
    #initialize the mean as weighted mean#
    mean <- sum(x * weights)/sum(weights);
mean.o<-mean.norMix(obj)
#initialize the variance as weighted variance#
    var <- sum(weights * (x - mean)^2)/sum(weights);
var.o<-var.norMix(obj)
    c(mean+mean.o, var/var.o)
}

#call library nor1mix#
library(nor1mix)

#deviance for norMix object: returning log-likelihood#
survreg.distributions$norMix$deviance<-function (y, scale, parms)
{
    #parms (parameters) consisting of the weights, means, and sd's#
    w<-parms[1:2];mu<-parms[3:4];sigma<-parms[5:6]
    #make norMix object#
    obj<-norMix(mu=mu,sig2=sigma^2,w=w)
    status <- y[, ncol(y)]
    width <- ifelse(status == 3, (y[, 2] - y[, 1])/scale, 0)
    #center for the object#
    center <- y[, 1] - width/2
```

```
    temp2 <- log(pnorMix(width/2,obj=obj,lower.tail=F)-
pnorMix(width/2,obj=obj))
#compute the loglikelihood for the object#
    best <- ifelse(status == 1, -log(dnorMix(0, obj = obj) * scale),
        ifelse(status == 3, temp2, 0))
    list(center = center, loglik = best)
}


#density of norMix object#
survreg.distributions$norMix$density<-function (x, parms)
{
      w<-parms[1:2];mu<-parms[3:4];sigma<-parms[5:6]
    #make norMix object consisting of weights, means, and sd's#
    obj<-norMix(mu=mu,sig2=sigma^2,w=w)
    #put in a vector the cdf, survival function, and the density#
     #for the norMix object#
    cbind(pnorMix(x, obj=obj), pnorMix(x, obj=obj,lower.tail=F),
dnorMix(x, obj=obj),
        -((w[1]/sigma[1]^3)*(x-mu[1])*dnorm((x-mu[1])/sigma[1]) +
          (w[2]/sigma[2]^3)*(x-mu[2])*dnorm((x-mu[2])/sigma[2]))/
dnorMix(x,obj=obj),
        ((w[1]/sigma[1]^3)*dnorm((x-mu[1])/sigma[1])*
((x-mu[1])^2/sigma[1]^2-1)
+       (w[2]/sigma[2]^3)*dnorm((x-mu[2])/sigma[2])*
```

```
((x-mu[2])^2/sigma[2]^2-1))/
dnorMix(x,obj=obj)
        )
}

#quantile function for norMix object#
survreg.distributions$norMix$quantile<-function (p, parms){
    w<-parms[1:2];mu<-parms[3:4];sigma<-parms[5:6]
  #make norMix object consisting of weights, means, and sd's#
  obj<-norMix(mu,sig2=sigma^2,w=w)
  #return the quantile for the norMix object#
  qnorMix(p, obj=obj)
}



  #Define Log norMix object in terms norMix object#
   #transformation: logarithm#
  #name attribute for Log normal mixture model#
survreg.distributions$lognorMix$name<-"Log norMix"
  #distribution is of type norMix#
survreg.distributions$lognorMix$dist<-survreg.distributions$norMix
  #transformation is logarithm#
survreg.distributions$lognorMix$trans<-function(y) log(y)
  #inverse transformation is exponential#
```

```
survreg.distributions$lognorMix$itrans<-function(x) exp(x)
  #derivative is 1/y#
survreg.distributions$lognorMix$dtrans<-function(y) 1/y
  #name lognorMix as log normal mixture model#
my.dist<-survreg.distributions$lognorMix
```

A.4.3 Sample Code for the Simulations Using the AFT Log-normal Mixture Model

For each simulation run, generate the gene expression data matrix X, the true survival times y_i's and censoring times c_i's (using Nguyen and Rocke's simulation procedure), and obtain the observed survival times $time1$, and censoring indicators $del1$. Using the 2-stage procedure, we apply dimension reduction methods (PCA, MPLS, RMPLS, RWPLS, RRWPLS, MIPLS, RMIPLS, UNIV, CPCR, SPCR) to X, and then apply the AFT model to the reduced data matrix ($scores$) in the second stage. Next, obtain the performance measures: mean square error of estimated weights on the genes ($MSE(beta)$), mean square error of fit ($MSE(fit)$), mean square error of the estimated survival function using the average of the covariates ($ave(ds)$), and mean square error of the estimated survival function using the covariates of the individuals ($ave(ds.ind)$). The procedure is repeated for 5000 simulations, and the performance measures are averaged over all the simulations.

```
#call library pls: pca, pls#
library(pls)
# Initialize number of simulations, #
 #N=number of patients#
 #p=number of genes#
```

```
 #d= number of underlying components for gene expressions#
 #vector r: in generation of gene expressions#
sim<-5000
N <- 50
p <- 100
d <- 6
r<-runif(d*N,-.5,0);dim(r)<-c(d,N)

#true regression parameters used in AFT model#
sd.pi <- .2
beta <- rnorm(p,0,sd.pi)

###-----------------------------------###
#True survival and censoring times#
#True survival: norm mixture: .9N(0,1)+.1N(0,5), #
 #censor: gamma:#
 #parameters for true survival and censoring times: #
   #sigma.T=1;shape.C=3,scale.C=0.16 (33% censor)#
mu<-0;sigma.T<-1;shape.C<-3;scale.C<-.16

#Initialize weights for the normal mixture w.c#
 #mean mu.c, sd^2 sig2.c#
w.c<-c(.9,.1);mu.c<-c(0,0);sig2.c<-c(1,100);
prms<-c(w.c,mu.c,sqrt(sig2.c))
 #define the object as norMix#
```

```
obj<-norMix(mu=mu.c,sig2 = sig2.c,w=w.c)

#Obtain the true censoring rate#
 #P[y_i>c_i] by conditioning on c_i#
integrand<-function(x,sigma.T,lambda.C){
pnorMix(x/sigma.T,obj=obj,lower.tail=F)*
dgamma(x,shape=shape.C,scale=scale.C)
}
 #true censoring rate given the various parameters#
(true.cens.rate<-integrate(integrand,lower=0,upper=Inf,
sigma.T=sigma.T,lambda.C=lambda.C)$value)
cens.rate<-0

#initial parameters used to generate covariate data#
 #matrix X#
mu_t <- 5/d;sd_t <- 1;
mu_e <- 0;sd_e <- .3;

#after choosing k's by cross-validation for each method:#
 #PCA: Principal Component Analysis#
 #MPLS: Modified Partial Least Squares#
 #RMPLS: Rank-based Modified Partial Least Squares#
 #RWPLS: Reweighted PLS#
 #RRWPLS: Rank-based Reweighted PLS#
 #MIPLS: Mean Imputation PLS#
```

```
 #RMIPLS: Rank-based Mean Imputation PLS#
 #CPCR: Correlation Principal Component Regression#
 #SPCR: Supervised Principal Component Regression#
 #UNIV: Univariate Selection#

k.pca<-5;k.mpls<-2;k.rmpls<-5;k.rwpls<-2;k.rrwpls<-1;
k.mipls<-2;k.rmipls<-5;
k.cpcr<-2;k.spcr<-1;k.univ<-6;


#-------Initialization-----------------#
#MSE(beta)#
MSE2.pca1<-rep(0,sim);
MSE2.mpls1<-MSE2.rmpls1<-MSE2.rmpls2<-MSE2.mipls1<-
MSE2.rmipls1<-MSE2.rwpls1<-MSE2.rrwpls1<-MSE2.pca1
MSE2.cpcr1<-MSE2.spcr1<-MSE2.univ1<-MSE2.pca1

#MSE(fit)#
fit.pca1<-fit.mpls1<-fit.rmpls1<-fit.rmpls2<-fit.rwpls1<-
fit.mipls1<-fit.rrwpls1<-fit.rmipls1<-rep(0,sim)
fit.univ1<-fit.cpcr1<-fit.spcr1<-fit.pca1

#ave(ds)#
ds.pca1<-ds.mpls1<-ds.rmpls1<-ds.rmpls2<-ds.rwpls1<-
ds.mipls1<-ds.rrwpls1<-ds.rmipls1<-rep(0,sim)
```

```
ds.univ1<-ds.cpcr1<-ds.spcr1<-ds.pca1

#ave(ds.ind)#
ds.pca1.ind<-ds.mpls1.ind<-ds.rmpls1.ind<-
ds.rwpls1.ind<-ds.mipls1.ind<-ds.rrwpls1.ind<-
ds.rmipls1.ind<-rep(0,sim)
ds.univ1.ind<-ds.cpcr1.ind<-ds.spcr1.ind<-ds.pca1.ind

#-------------------------------------------------#
#-------START FOR loop-------------------------------#
#-------------------------------------------------#
for(i in 1:sim){
  #underlying components for data matrix X#
  tau.p<-matrix(rnorm(d*p,mu_t, sd_t),nrow=d,ncol=p);
  #error component for data matrix X#
  e <- matrix(rnorm(N*p, mu_e, sd_e),nrow=N,ncol=p);

  #--------------Gene-Expression matrix X-------------
  x1<-exp(t(r) %*% tau.p + e);x.m<-colMeans(x1);
  x.sd<-apply(x1,2,sd)
  #standardize the data matrix X (mean 0, variance 1#
  # for each column#
  x1<-(x1)/rep(x.sd,each=N)

  #--------------Survival times-----------------------
```

```
  ### Survival Times ###
    #censoring rate = 1/3#
  #Error: True: lognormal, censor: gamma: sigma.T=1;
shape.C=3, scale.C=0.685 (33% censor)#

  #Generate: true errors e.i~.9*N(0,1)+.1*N(0,10^2)#
  e.i<-rnorMix(n=N,obj=obj)
  #Generate: censor errors w.i~Gamma(shape,scale)#
  w.i<-rgamma(n=N,shape=shape.C,scale=scale.C)
  #true survival times for AFT model#
  y1<-exp(mu+x1%*%beta+sigma.T*e.i)
  #censoring times for AFT model#
  z1<-exp(mu+x1%*%beta+w.i)
  #observed survival times: minimum of true & #
   #censoring times#
  time1 <- pmin(y1,z1)
  #censoring indicators#
  del1 <- ifelse(y1<z1,1,0)
  #observed censoring rate#
  cens.rate<- sum(cens.rate,(N-sum(del1))/N)

  #Reweighted and Mean Imputation for censored response###
  # both nonparametric methods are from Datta et al.##
   #adjust surv. times using reweighted#
  r.y1<-rw(time1,del1)$y.t
```

```
   #adjust surv. times using mean imputation#
  mi.y1<-mi(time1,del1)$y.t

   #_____________________________________________#
  ###### PCA, MPLS and RMPLS########
   #obtain weight vectors and components#
   #function pcr: from library pls#
   #function nipals: from function for procedure RMPLS#
   #ncomp: number of components#
   #no cross-validation performed#
   #my.dist: AFT lognormal mixture model#
  sim.pca1<-pcr(time1 ~ x1, ncomp=k.pca, validation = "none")
  sim.mpls1<-nipals(x1,time1, del1,  ncomp=k.mpls,
cor.method=1,method=4,my.dist=my.dist,parms=prms)
  sim.rmpls1<-nipals(x1,time1, del1,  ncomp=k.rmpls,
cor.method=2,method=4,my.dist=my.dist,parms=prms)
  sim.pca1.loads<-sim.pca1$loadings[,1:k.pca]
sim.pca1.scores<-x1%*%sim.pca1.loads
  sim.mpls1.loads<-sim.mpls1$loadings[,1:k.mpls]
sim.mpls1.scores<-x1%*%sim.mpls1.loads
  sim.rmpls1.loads<-sim.rmpls1$loadings[,1:k.rmpls]
sim.rmpls1.scores<-x1%*%sim.rmpls1.loads

  #-------------------------------------------------
  #Reweighted, Mean Imputation for censoring in PLS: #
```

```
   #the usual and Rank-based#
   #del=0: no censoring (the censoring already taken #
    #into account when adjust for the survival times#
sim.rwpls1<-nipals(x1,r.y1, del=0, ncomp=k.rwpls,
cor.method=1)
  sim.mipls1<-nipals(x1,mi.y1, del=0, ncomp=k.mipls,
cor.method=1)
  sim.rrwpls1<-nipals(x1,r.y1, del=0, ncomp=k.rrwpls,
cor.method=2)
  sim.rmipls1<-nipals(x1,mi.y1, del=0, ncomp=k.rmipls,
cor.method=2)

   #obtain weight vectors and components#
  sim.rwpls1.loads<-sim.rwpls1$loadings[,1:k.rwpls]
  sim.rwpls1.scores<-x1%*%sim.rwpls1.loads
  sim.mipls1.loads<-sim.mipls1$loadings[,1:k.mipls]
  sim.mipls1.scores<-x1%*%sim.mipls1.loads
  sim.rrwpls1.loads<-sim.rrwpls1$loadings[,1:k.rrwpls]
  sim.rrwpls1.scores<-x1%*%sim.rrwpls1.loads
  sim.rmipls1.loads<-sim.rmipls1$loadings[,1:k.rmipls]
  sim.rmipls1.scores<-x1%*%sim.rmipls1.loads

  #------Univariate Selection (UNIV) and Supervised PCR  #
  # --for SPCR: first select 20% of genes, then use UNIV ##
   # SPCR: Bair and Tibshirani, UNIV: Bolvestad#
```

```
sim.univ1<-univ(x1,time1,del1,ncomp=max(k.univ,k.spcr),
lambda=.2,method=2,dis=4,my.dist=my.dist,parms=prms)
  sim.univ1.loads<-sim.univ1$univ.loads[,1:k.univ]
  sim.univ1.scores<-sim.univ1$univ.scores[,1:k.univ]
sim.spcr1.loads<-sim.univ1$spcr.loads[,1:k.spcr]
sim.spcr1.scores<-sim.univ1$spcr.scores[,1:k.spcr]

  #--------------------Correlation PCA----------------------#
  # CPCR: Sun and Zao #
  sim.cpcr1<-cpcr(x1,time1,del1,ncomp=k.cpcr,method=2,dis=4,
my.dist=my.dist,parms=prms)
sim.cpcr1.loads<-matrix(sim.cpcr1$loadings[,1:k.cpcr],
ncol=k.cpcr);
  sim.cpcr1.scores<-x1%*%sim.cpcr1.loads

#_____________________________________________________#
#AFT Model: lognormal mixture model ####
##after dimension reduction##
# use survreg in library survival for AFT model#
# specify distribution (dist), with parameters (parms)#
# my.dist is lognormal-mixture object#
aft.pca1<-survreg(Surv(time1,del1)~sim.pca1.scores,dist=
my.dist,parms=prms)
aft.rwpls1<-survreg(Surv(time1,del1)~sim.rwpls1.scores,dist=
my.dist,parms=prms)
```

```
aft.mipls1<-survreg(Surv(time1,del1)~sim.mipls1.scores,dist=
my.dist,parms=prms)
aft.rrwpls1<-survreg(Surv(time1,del1)~sim.rrwpls1.scores,dist=
my.dist,parms=prms)
aft.rmipls1<-survreg(Surv(time1,del1)~sim.rmipls1.scores,dist=
my.dist,parms=prms)
aft.mpls1<-survreg(Surv(time1,del1)~sim.mpls1.scores,dist=
my.dist,parms=prms)
aft.rmpls1<-survreg(Surv(time1,del1)~sim.rmpls1.scores,dist=
my.dist,parms=prms)
aft.cpcr1<-survreg(Surv(time1,del1)~sim.cpcr1.scores,dist=
my.dist,parms=prms)
aft.spcr1<-survreg(Surv(time1,del1)~sim.spcr1.scores,dist=
my.dist,parms=prms)
aft.univ1<-survreg(Surv(time1,del1)~sim.univ1.scores,dist=
my.dist,parms=prms)

#extract the coefficients from AFT model#
pca1.coef<-aft.pca1$coef;
rwpls1.coef<-aft.rwpls1$coef; mipls1.coef<-aft.mipls1$coef
rrwpls1.coef<-aft.rrwpls1$coef; rmipls1.coef<-aft.rmipls1$coef;
mpls1.coef<-aft.mpls1$coef;rmpls1.coef<-aft.rmpls1$coef;
cpcr1.coef<-aft.cpcr1$coef; spcr1.coef<-aft.spcr1$coef;
univ1.coef<-aft.univ1$coef;
```

```
#_____________________________________________________#
###true survival### plnorm(est.time1[[i]],#
 #meanlog=pca1.coef[1,,i]+apply(x1[,,i],2,mean) %*% #
 #sim.pca1.loads[,,i]%*%pca1.coef[-1,,i],sdlog=sigma.T,#
 #lower.tail=F)#

 #estimated survival at average of covariates#
  log.est.time1<-log(sort(subset(time1,del1==1)))
  true.surv1<-pnorMix(q=log.est.time1-mu-apply(x1,2,mean) %*%
beta,obj=obj,lower.tail=F)

  #using cdf of norMix dist.: implemented in survreg.distributions#
  surv.pca1<-pnorMix(q=log.est.time1-pca1.coef[1]-
apply(x1,2,mean) %*% sim.pca1.loads%*%pca1.coef[-1],
obj=obj,lower.tail=F)
  surv.rwpls1<-pnorMix(q=log.est.time1-rwpls1.coef[1]-
apply(x1,2,mean) %*% sim.rwpls1.loads%*%rwpls1.coef[-1],
obj=obj,lower.tail=F)
surv.mipls1<-pnorMix(q=log.est.time1-mipls1.coef[1]-
apply(x1,2,mean) %*% sim.mipls1.loads%*%mipls1.coef[-1],
obj=obj,lower.tail=F)
  surv.rrwpls1<-pnorMix(q=log.est.time1-rrwpls1.coef[1]-
apply(x1,2,mean) %*% sim.rrwpls1.loads%*%rrwpls1.coef[-1],
obj=obj,lower.tail=F)
surv.rmipls1<-pnorMix(q=log.est.time1-rmipls1.coef[1]-
```

```
apply(x1,2,mean) %*% sim.rmipls1.loads%*%rmipls1.coef[-1],
obj=obj,lower.tail=F)
  surv.mpls1<-pnorMix(q=log.est.time1-mpls1.coef[1]-
apply(x1,2,mean) %*% sim.mpls1.loads%*%mpls1.coef[-1],
obj=obj,lower.tail=F)
  surv.rmpls1<-pnorMix(q=log.est.time1-rmpls1.coef[1]-
apply(x1,2,mean) %*% sim.rmpls1.loads%*%rmpls1.coef[-1],
obj=obj,lower.tail=F)
  surv.cpcr1<-pnorMix(q=log.est.time1-cpcr1.coef[1]-
apply(x1,2,mean) %*% sim.cpcr1.loads%*%cpcr1.coef[-1],
obj=obj,lower.tail=F)
  surv.spcr1<-pnorMix(q=log.est.time1-spcr1.coef[1]-
apply(x1,2,mean) %*% sim.spcr1.loads%*%spcr1.coef[-1],
obj=obj,lower.tail=F)
  surv.univ1<-pnorMix(q=log.est.time1-univ1.coef[1]-
apply(x1,2,mean) %*% sim.univ1.loads%*%univ1.coef[-1],
obj=obj,lower.tail=F)

#-------------------------------------------------#
#True survival#
#estimated survival: individual covariates#
true.surv1.ind<-matrix(0,N,length(log.est.time1))
surv.pca1.ind<-surv.rwpls1.ind<-surv.mipls1.ind<-
surv.rrwpls1.ind<-surv.rmipls1.ind<-true.surv1.ind
surv.mpls1.ind<-surv.rmpls1.ind<-true.surv1.ind
```

```
surv.cpcr1.ind<-surv.spcr1.ind<-surv.univ1.ind<-
true.surv1.ind

  for(j in 1:N){
    true.surv1.ind[j,]<-pnorMix(log.est.time1-mu-x1[j,]%*%
beta,obj=obj,lower.tail=F)
    surv.pca1.ind[j,]<-pnorMix(log.est.time1-pca1.coef[1]-
x1[j,]%*%sim.pca1.loads%*%pca1.coef[-1],obj=obj,lower.tail=F)
    surv.rwpls1.ind[j,]<-pnorMix(log.est.time1-rwpls1.coef[1]-
x1[j,]%*%sim.rwpls1.loads%*%rwpls1.coef[-1],obj=obj,
lower.tail=F)
    surv.mipls1.ind[j,]<-pnorMix(log.est.time1-mipls1.coef[1]-
x1[j,]%*%sim.mipls1.loads%*%mipls1.coef[-1],obj=obj,
lower.tail=F)
    surv.rrwpls1.ind[j,]<-pnorMix(log.est.time1-rrwpls1.coef[1]-
x1[j,]%*%sim.rrwpls1.loads%*%rrwpls1.coef[-1],obj=obj,
lower.tail=F)
    surv.rmipls1.ind[j,]<-pnorMix(log.est.time1-rmipls1.coef[1]-
x1[j,]%*%sim.rmipls1.loads%*%rmipls1.coef[-1],obj=obj,
lower.tail=F)
    surv.mpls1.ind[j,]<-pnorMix(log.est.time1-mpls1.coef[1]-
x1[j,]%*%sim.mpls1.loads%*%mpls1.coef[-1],obj=obj,
lower.tail=F)
    surv.rmpls1.ind[j,]<-pnorMix(log.est.time1-rmpls1.coef[1]-
x1[j,]%*%sim.rmpls1.loads%*%rmpls1.coef[-1],obj=obj,
```

```
lower.tail=F)
    surv.cpcr1.ind[j,]<-pnorMix(log.est.time1-cpcr1.coef[1]-
x1[j,]%*%sim.cpcr1.loads%*%cpcr1.coef[-1],obj=obj,
lower.tail=F)
    surv.spcr1.ind[j,]<-pnorMix(log.est.time1-spcr1.coef[1]-
x1[j,]%*%sim.spcr1.loads%*%spcr1.coef[-1],obj=obj,
lower.tail=F)
    surv.univ1.ind[j,]<-pnorMix(log.est.time1-univ1.coef[1]-
x1[j,]%*%sim.univ1.loads%*%univ1.coef[-1],obj=obj,
lower.tail=F)
  }

  ###obtaining the MSE(betas)###
MSE2.pca1[i]<-sum((beta-sim.pca1.loads%*%pca1.coef[-1])^2)
MSE2.rwpls1[i]<-sum((beta-matrix(sim.rwpls1.loads,
ncol=k.rwpls)%*%rwpls1.coef[-1])^2)
MSE2.mipls1[i]<-sum((beta-matrix(sim.mipls1.loads,
ncol=k.mipls)%*%mipls1.coef[-1])^2)
MSE2.rrwpls1[i]<-sum((beta-matrix(sim.rrwpls1.loads,
ncol=k.rrwpls)%*%rrwpls1.coef[-1])^2)
MSE2.rmipls1[i]<-sum((beta-matrix(sim.rmipls1.loads,
ncol=k.rmipls)%*%rmipls1.coef[-1])^2)
MSE2.mpls1[i]<-sum((beta-matrix(sim.mpls1.loads,
ncol=k.mpls)%*%mpls1.coef[-1])^2)
MSE2.rmpls1[i]<-sum((beta-matrix(sim.rmpls1.loads,
```

```
ncol=k.rmpls)%*%rmpls1.coef[-1])^2)
MSE2.cpcr1[i]<-sum((beta-matrix(sim.cpcr1.loads,
ncol=k.cpcr)%*%cpcr1.coef[-1])^2)
MSE2.spcr1[i]<-sum((beta-matrix(sim.spcr1.loads,
ncol=k.spcr)%*%spcr1.coef[-1])^2)
MSE2.univ1[i]<-sum((beta-matrix(sim.univ1.loads,
ncol=k.univ)%*%univ1.coef[-1])^2)

    ###obtaining the MSE(fit)###
  fit.pca1[i]<-sum((pca1.coef[1]+matrix(sim.pca1.scores,
ncol=k.pca)%*%pca1.coef[-1]-log(time1))^2*del1)/sum(del1)
fit.rwpls1[i]<-sum((rwpls1.coef[1]+matrix(sim.rwpls1.scores,
ncol=k.rwpls)%*%rwpls1.coef[-1]-log(time1))^2*del1)/sum(del1)
fit.mipls1[i]<-sum((mipls1.coef[1]+matrix(sim.mipls1.scores,
ncol=k.mipls)%*%mipls1.coef[-1]-log(time1))^2*del1)/sum(del1)
fit.rrwpls1[i]<-sum((rrwpls1.coef[1]+matrix(sim.rrwpls1.scores,
ncol=k.rrwpls)%*%rrwpls1.coef[-1]-log(time1))^2*del1)/sum(del1)
fit.rmipls1[i]<-sum((rmipls1.coef[1]+matrix(sim.rmipls1.scores,
ncol=k.rmipls)%*%rmipls1.coef[-1]-log(time1))^2*del1)/sum(del1)
fit.mpls1[i]<-sum((mpls1.coef[1]+matrix(sim.mpls1.scores,
ncol=k.mpls)%*%mpls1.coef[-1]-log(time1))^2*del1)/sum(del1)
fit.rmpls1[i]<-sum((rmpls1.coef[1]+matrix(sim.rmpls1.scores,
ncol=k.rmpls)%*%rmpls1.coef[-1]-log(time1))^2*del1)/sum(del1)
fit.cpcr1[i]<-sum((cpcr1.coef[1]+matrix(sim.cpcr1.scores,
ncol=k.cpcr)%*%cpcr1.coef[-1]-log(time1))^2*del1)/sum(del1)
```

```
fit.spcr1[i]<-sum((spcr1.coef[1]+matrix(sim.spcr1.scores,
ncol=k.spcr)%*%spcr1.coef[-1]-log(time1))^2*del1)/sum(del1)
fit.univ1[i]<-sum((univ1.coef[1]+matrix(sim.univ1.scores,
ncol=k.univ)%*%univ1.coef[-1]-log(time1))^2*del1)/sum(del1)

  #_____________________________________________________#
  ###Squared Euclidean distance for average individual###
  ds.pca1[i]<-sum((surv.pca1-true.surv1)^2)
ds.rwpls1[i]<-sum((surv.rwpls1-true.surv1)^2)
ds.mipls1[i]<-sum((surv.mipls1-true.surv1)^2)
ds.rrwpls1[i]<-sum((surv.rrwpls1-true.surv1)^2)
ds.rmipls1[i]<-sum((surv.rmipls1-true.surv1)^2)
ds.mpls1[i]<-sum((surv.mpls1-true.surv1)^2)
ds.rmpls1[i]<-sum((surv.rmpls1-true.surv1)^2)

ds.cpcr1[i]<-sum((surv.cpcr1-true.surv1)^2)
ds.univ1[i]<-sum((surv.univ1-true.surv1)^2)
ds.spcr1[i]<-sum((surv.spcr1-true.surv1)^2)

  #---------------------------------------###
  #--Squared Euclidean distance using individual covariates##
  #obs.ave(d^2.ind1) = (1/sim)*sum{i=1,sim} (1/N*sum{n=1,N}
[sum{t in D} (S_{in}(t) - S.h_{in}(t))^2))])
  ds.pca1.ind[i]<-mean(apply((surv.pca1.ind-
true.surv1.ind)^2,1,sum))
```

```
ds.rwpls1.ind[i]<-mean(apply((surv.rwpls1.ind-
true.surv1.ind)^2,1,sum))
ds.mipls1.ind[i]<-mean(apply((surv.mipls1.ind-
true.surv1.ind)^2,1,sum))
ds.rrwpls1.ind[i]<-mean(apply((surv.rrwpls1.ind-
true.surv1.ind)^2,1,sum))
ds.rmipls1.ind[i]<-mean(apply((surv.rmipls1.ind-
true.surv1.ind)^2,1,sum))
ds.mpls1.ind[i]<-mean(apply((surv.mpls1.ind-
true.surv1.ind)^2,1,sum))
ds.rmpls1.ind[i]<-mean(apply((surv.rmpls1.ind-
true.surv1.ind)^2,1,sum))
ds.cpcr1.ind[i]<-mean(apply((surv.cpcr1.ind-
true.surv1.ind)^2,1,sum))
ds.spcr1.ind[i]<-mean(apply((surv.spcr1.ind-
true.surv1.ind)^2,1,sum))
ds.univ1.ind[i]<-mean(apply((surv.univ1.ind-
true.surv1.ind)^2,1,sum))
}
#-------------END FOR LOOP-------------------------------#
####average the performance measures for all simulations & #
###obtain the standard error for the average###
#####################################################
#MSE(beta)#
(MSE2.pca1.mean<-mean(MSE2.pca1,na.rm=T))
```

```
(MSE2.rwpls1.mean<-mean(MSE2.rwpls1,na.rm=T))
(MSE2.mipls1.mean<-mean(MSE2.mipls1,na.rm=T))
(MSE2.rrwpls1.mean<-mean(MSE2.rrwpls1,na.rm=T))
(MSE2.rmipls1.mean<-mean(MSE2.rmipls1,na.rm=T))
(MSE2.mpls1.mean<-mean(MSE2.mpls1,na.rm=T))
(MSE2.rmpls1.mean<-mean(MSE2.rmpls1,na.rm=T))
(MSE2.cpcr1.mean<-mean(MSE2.cpcr1,na.rm=T))
(MSE2.spcr1.mean<-mean(MSE2.spcr1,na.rm=T))
(MSE2.univ1.mean<-mean(MSE2.univ1,na.rm=T))

(MSE2.pca1.sd<-sd(MSE2.pca1,na.rm=T))
(MSE2.mpls1.sd<-sd(MSE2.mpls1,na.rm=T))
(MSE2.rmpls1.sd<-sd(MSE2.rmpls1,na.rm=T))
(MSE2.rmpls2.sd<-sd(MSE2.rmpls2,na.rm=T))
(MSE2.mipls1.sd<-sd(MSE2.mipls1,na.rm=T))
(MSE2.rmipls1.sd<-sd(MSE2.rmipls1,na.rm=T))
(MSE2.rwpls1.sd<-sd(MSE2.rwpls1,na.rm=T))
(MSE2.rrwpls1.sd<-sd(MSE2.rrwpls1,na.rm=T))
(MSE2.cpcr1.sd<-sd(MSE2.cpcr1,na.rm=T))
(MSE2.spcr1.sd<-sd(MSE2.spcr1,na.rm=T))
(MSE2.univ1.sd<-sd(MSE2.univ1,na.rm=T))

#MSE(fit)#
(fit.pca1.mean<-mean(fit.pca1,na.rm=T))
(fit.rwpls1.mean<-mean(fit.rwpls1,na.rm=T))
```

```
(fit.mipls1.mean<-mean(fit.mipls1,na.rm=T))
(fit.rrwpls1.mean<-mean(fit.rrwpls1,na.rm=T))
(fit.rmipls1.mean<-mean(fit.rmipls1,na.rm=T))
(fit.mpls1.mean<-mean(fit.mpls1,na.rm=T))
(fit.rmpls1.mean<-mean(fit.rmpls1,na.rm=T))
(fit.cpcr1.mean<-mean(fit.cpcr1,na.rm=T))
(fit.spcr1.mean<-mean(fit.spcr1,na.rm=T))
(fit.univ1.mean<-mean(fit.univ1,na.rm=T))

(fit.pca1.sd<-sd(fit.pca1,na.rm=T))
(fit.mpls1.sd<-sd(fit.mpls1,na.rm=T))
(fit.rmpls1.sd<-sd(fit.rmpls1,na.rm=T))
(fit.rmpls2.sd<-sd(fit.rmpls2,na.rm=T))
(fit.mipls1.sd<-sd(fit.mipls1,na.rm=T))
(fit.rmipls1.sd<-sd(fit.rmipls1,na.rm=T))
(fit.rwpls1.sd<-sd(fit.rwpls1,na.rm=T))
(fit.rrwpls1.sd<-sd(fit.rrwpls1,na.rm=T))
(fit.cpcr1.sd<-sd(fit.cpcr1,na.rm=T))
(fit.spcr1.sd<-sd(fit.spcr1,na.rm=T))
(fit.univ1.sd<-sd(fit.univ1,na.rm=T))

#ave(ds)#
(ds.pca1.mean<-mean(ds.pca1,na.rm=T))
(ds.rwpls1.mean<-mean(ds.rwpls1,na.rm=T))
(ds.mipls1.mean<-mean(ds.mipls1,na.rm=T))
```

```
(ds.rrwpls1.mean<-mean(ds.rrwpls1,na.rm=T))
(ds.rmipls1.mean<-mean(ds.rmipls1,na.rm=T))
(ds.mpls1.mean<-mean(ds.mpls1,na.rm=T))
(ds.rmpls1.mean<-mean(ds.rmpls1,na.rm=T))
(ds.cpcr1.mean<-mean(ds.cpcr1,na.rm=T))
(ds.spcr1.mean<-mean(ds.spcr1,na.rm=T))
(ds.univ1.mean<-mean(ds.univ1,na.rm=T))

(ds.pca1.sd<-sd(ds.pca1,na.rm=T))
(ds.rwpls1.sd<-sd(ds.rwpls1,na.rm=T))
(ds.mipls1.sd<-sd(ds.mipls1,na.rm=T))
(ds.rrwpls1.sd<-sd(ds.rrwpls1,na.rm=T))
(ds.rmipls1.sd<-sd(ds.rmipls1,na.rm=T))
(ds.mpls1.sd<-sd(ds.mpls1,na.rm=T))
(ds.rmpls1.sd<-sd(ds.rmpls1,na.rm=T))
(ds.cpcr1.sd<-sd(ds.cpcr1,na.rm=T))
(ds.spcr1.sd<-sd(ds.spcr1,na.rm=T))
(ds.univ1.sd<-sd(ds.univ1,na.rm=T))

#ave(ds.ind)#
(ds.pca1.ind.ave<-mean(ds.pca1.ind))
(ds.rwpls1.ind.ave<-mean(ds.rwpls1.ind))
(ds.mipls1.ind.ave<-mean(ds.mipls1.ind))
(ds.rrwpls1.ind.ave<-mean(ds.rrwpls1.ind))
(ds.rmipls1.ind.ave<-mean(ds.rmipls1.ind))
```

```
(ds.mpls1.ind.ave<-mean(ds.mpls1.ind))
(ds.rmpls1.ind.ave<-mean(ds.rmpls1.ind))
(ds.cpcr1.ind.ave<-mean(ds.cpcr1.ind))
(ds.spcr1.ind.ave<-mean(ds.spcr1.ind))
(ds.univ1.ind.ave<-mean(ds.univ1.ind))

(ds.pca1.ind.sd<-sd(ds.pca1.ind))
(ds.rwpls1.ind.sd<-sd(ds.rwpls1.ind))
(ds.mipls1.ind.sd<-sd(ds.mipls1.ind))
(ds.rrwpls1.ind.sd<-sd(ds.rrwpls1.ind))
(ds.rmipls1.ind.sd<-sd(ds.rmipls1.ind))
(ds.mpls1.ind.sd<-sd(ds.mpls1.ind))
(ds.rmpls1.ind.sd<-sd(ds.rmpls1.ind))
(ds.cpcr1.ind.sd<-sd(ds.cpcr1.ind))
(ds.spcr1.ind.sd<-sd(ds.spcr1.ind))
(ds.univ1.ind.sd<-sd(ds.univ1.ind))
```

A.4.4 Sample Code for the Cross-Validation to Select k Using the AFT Lognormal Mixture Model

For each method of dimension reduction, we apply the 2-stage procedure. The selection of k is based on the minimization of the cross-validation squared error of fit under the AFT model. The idea is after the dimension reduction stage, we obtain a reduced data matrix. We split the reduced data matrix into a training and a test set, and use the training set to obtain estimates of the coefficients under the AFT model, and use these estimated coefficients to validate the test set. The following code provides the

cross-validation procedure to be incorporated with the previous subsection.

```
##outside the for loop, add the following for initialization##
K<-20; k.s<-1:K  #iniatialize K for various methods#
K.mpls<-20;k.s2<-1:K.mpls
K.mipls<-20;k.s2b<-1:K.mipls
K.univ<-20;k.s3<-1:K.univ
K.spcr<-20;k.s4<-1:K.spcr
K.max<-max(K,K.mpls,K.mipls,K.univ,K.spcr);k.m<-1:K.max

#initialize the CV(fit)#
CV.pca1.fit.ave<-matrix(0,sim,length(k.s));
CV.mpls1.fit.ave<-CV.rmpls1.fit.ave<-matrix(0,sim,length(k.s2))
CV.rwpls1.fit.ave<-CV.rrwpls1.fit.ave<-matrix(0,sim,length(k.s2b))
CV.mipls1.fit.ave<-CV.rmipls1.fit.ave<-matrix(0,sim,length(k.s2b))
CV.univ1.fit.ave<-matrix(0,sim,length(k.s3))
CV.cpcr1.fit.ave<-CV.spcr1.fit.ave<-matrix(0,sim,length(k.s4))

   #######in the for loop####
    #after generating gene expression data X and response y#
     #initialize for CV for each fold#
       CV.pca1.fit<-matrix(0,floor(N/nfold),length(k.s));
  CV.mpls1.fit<-CV.rmpls1.fit<-matrix(0,floor(N/nfold),
length(k.s2))
  CV.rwpls1.fit<-CV.rrwpls1.fit<-matrix(0,floor(N/nfold),
length(k.s2b))
```

```
  CV.mipls1.fit<-CV.rmipls1.fit<-matrix(0,floor(N/nfold),
length(k.s2b))
  CV.univ1.fit<-matrix(0,floor(N/nfold),length(k.s3))
  CV.cpcr1.fit<-CV.spcr1.fit<-matrix(0,floor(N/nfold),
length(k.s4))

  n.seq<-1:nrow(x1)
 #Cross validation#
  for (fold in 1:floor(N/nfold)) {
    counter<-T
    while(counter){
      #randomly select a test vector#
      test <- sample(n.seq,nfold,replace=F)
      counter<-(sum(del1[test])==0)
    }
    #extract the training and test sets using the #
     #vector#
    x1.train<-x1[-test,];time1.train<-time1[-test];
del1.train<-del1[-test];N.train<-length(time1.train)
x1.test<-x1[test,];time1.test<-time1[test];
del1.test<-del1[test];N.test<-length(time1.test)
    n.seq<-n.seq[-test]

  #The loadings and scores vectors for each method are obtained for#
   #training set#
```

```
   #for example: for PCA#
   sim.pca1<-pcr(time1.train ~ x1.train, ncomp=K, validation = "none")
   sim.pca1.loads<-sim.pca1$loadings[,1:K]
sim.pca1.scores<-x1.train%*%sim.pca1.loads

  ##Cross-validation of fit error##
    #initialization#
  pca1.coef<-list(0); #coef from AFT model#
  mpls1.coef<-rmpls1.coef<-pca1.coef
  rwpls1.coef<-rrwpls1.coef<-pca1.coef
  mipls1.coef<-rmipls1.coef<-pca1.coef
  cpcr1.coef<-spcr1.coef<-univ1.coef<-pca1.coef

  pca1.fit.err<-rep(0,length(k.s)); #squared fit error#
  mpls1.fit.err<-rmpls1.fit.err<-rep(0,length(k.s2));
  rwpls1.fit.err<-rrwpls1.fit.err<-rep(0,length(k.s2b));
  mipls1.fit.err<-rmipls1.fit.err<-rep(0,length(k.s2b));
  univ1.fit.err<-rep(0,length(k.s3));
  cpcr1.fit.err<-spcr1.fit.err<-rep(0,length(k.s4))

   #for each method, obtain the CV fit error#
   # for example: PCA#
  for(j in 1:length(k.m)){
    #PCA
    if(j <= K){     aft.pca1<-try(survreg(Surv(time1.train,
```

```
del1.train)~sim.pca1.scores[,1:j],dist=my.dist,parms=prms),T)
      #coefficients from AFT model#
    coef.tmp<-try(aft.pca1$coef,T)
      #use est. coef. from train set to estimated #
        #y.hat for test set#
    #CV(fit.error)#
    ifelse(j==1,
      pca1.fit.err[j]<-sum((matrix(sim.pca2.scores[,1:j],
ncol=1)%*%coef.tmp[-1] + coef.tmp[1]-log(time1.test))^2*
del1.test)/sum(del1.test),
      pca1.fit.err[j]<-sum((sim.pca2.scores[,1:j]%*%
coef.tmp[-1] + coef.tmp[1]-log(time1.test))^2*del1.test)/
sum(del1.test)
    )
   }

  #Accumulate the fit error after each fold#
   #for example: PCA#
     CV.pca1.fit[fold,]<-try(CV.pca1.fit[fold,] +
pca1.fit.err,T)

  #average over all the simulations#
   #for example: PCA#
  CV.pca1.fit.ave[i,]<-apply(CV.pca1.fit,2,mean,na.rm=T)
```

Appendix B

Appendix: Comparison Plots of the Different Methods

B.1 Simulation: Cox Model

B.1.1 Scenario 1: Fix $k = 3$

Figure B.1 : Cox model: 1/3 censoring with $p = 100$ and $p = 1000$ for one simulation run. The observed survival times $T_i = min(y_i, c_i)$ are plotted against $X_i'\beta$, where $i = 1, \ldots, N$.

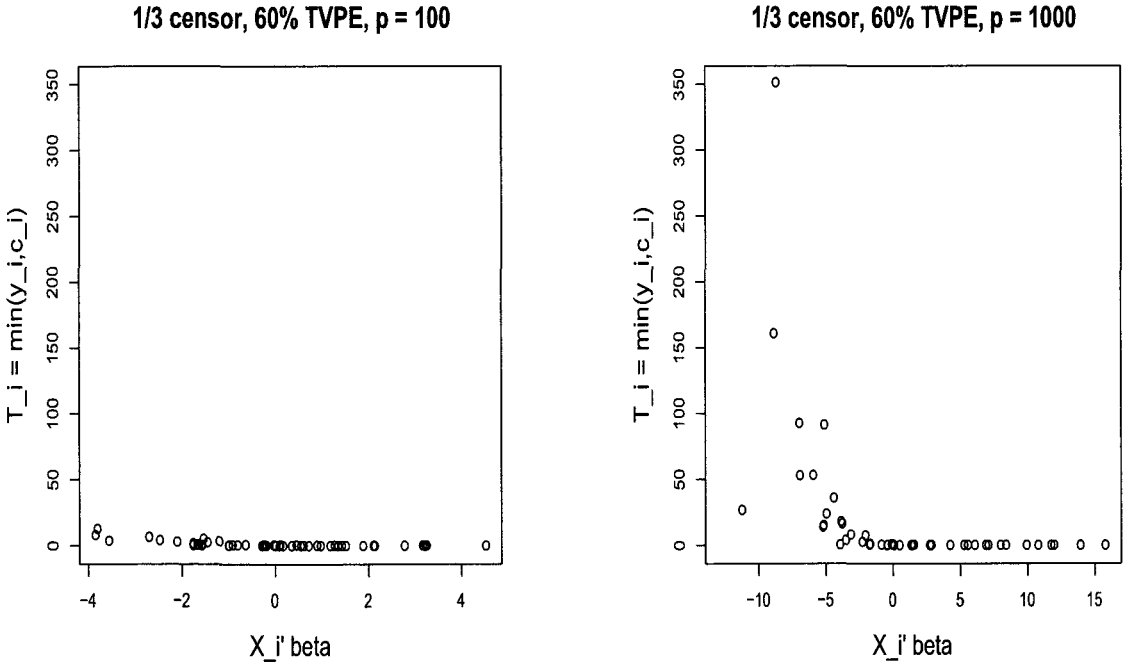


Figure B.2 : Cox model: 1/3 censored. The $ave(bias)$ of survival (using the average of the covariates) is plotted against q, quantiles of the true survival, for datasets with 50%, 60% and 70% TVPE accounted for by the first 3 PCs comparing PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV. The $x-$axis denotes q, and the y-axis denotes $ave(bias)$. The rows of the plots are for datasets with dimension $p = 100, 500$, and 800.

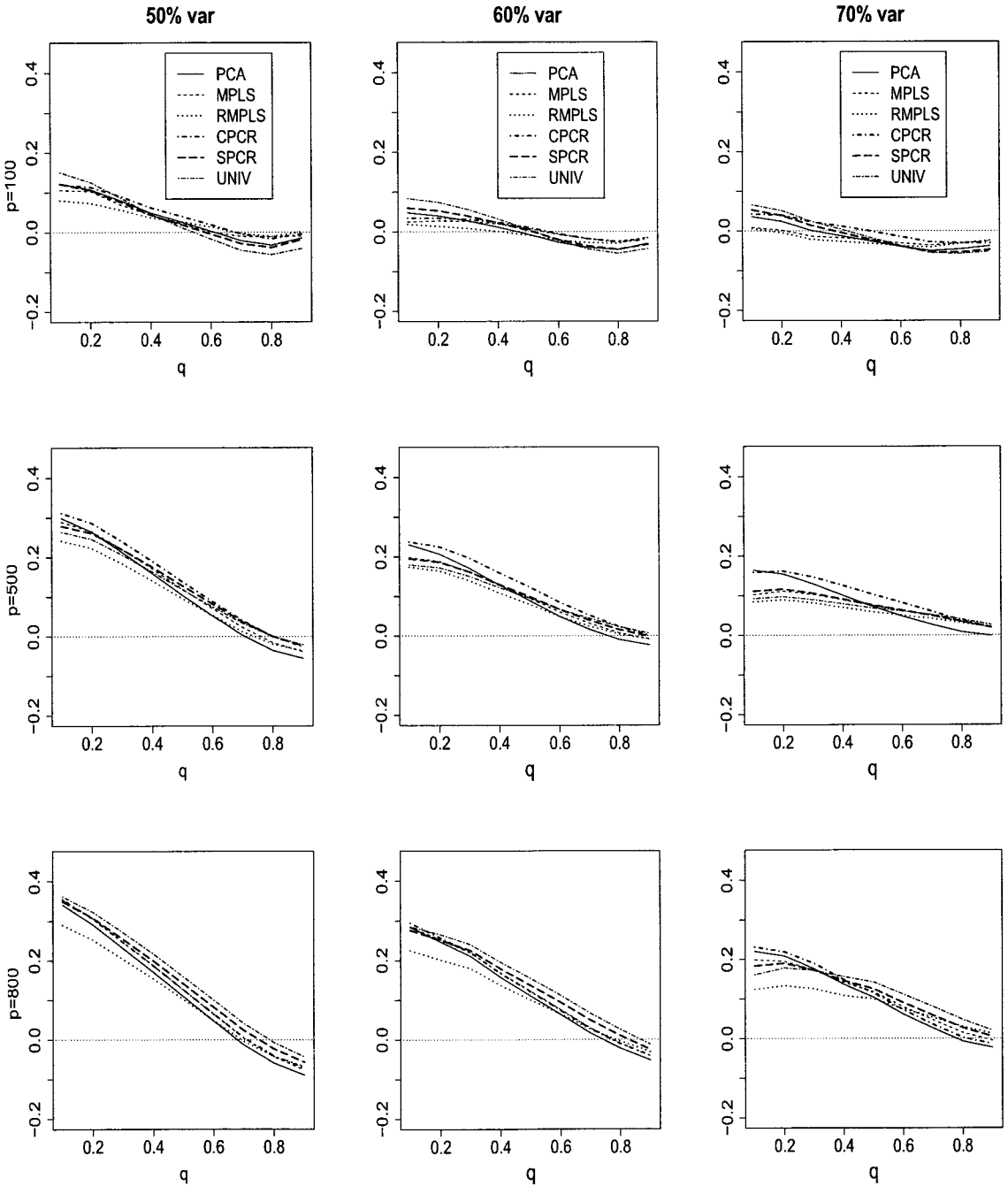


Figure B.3 : Cox model: 1/3 censored. The *ave(bias.ind)* of survival (using the covariates of the individuals) is plotted against q, quantiles of the true survival, for datasets with 50%, 60% and 70% TVPE accounted for by the first 3 PCs comparing PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV. The $x-$axis denotes q, and the y-axis denotes *ave(bias.ind)*. The rows of the plots are for datasets with dimension $p = 100, 500$, and 800.

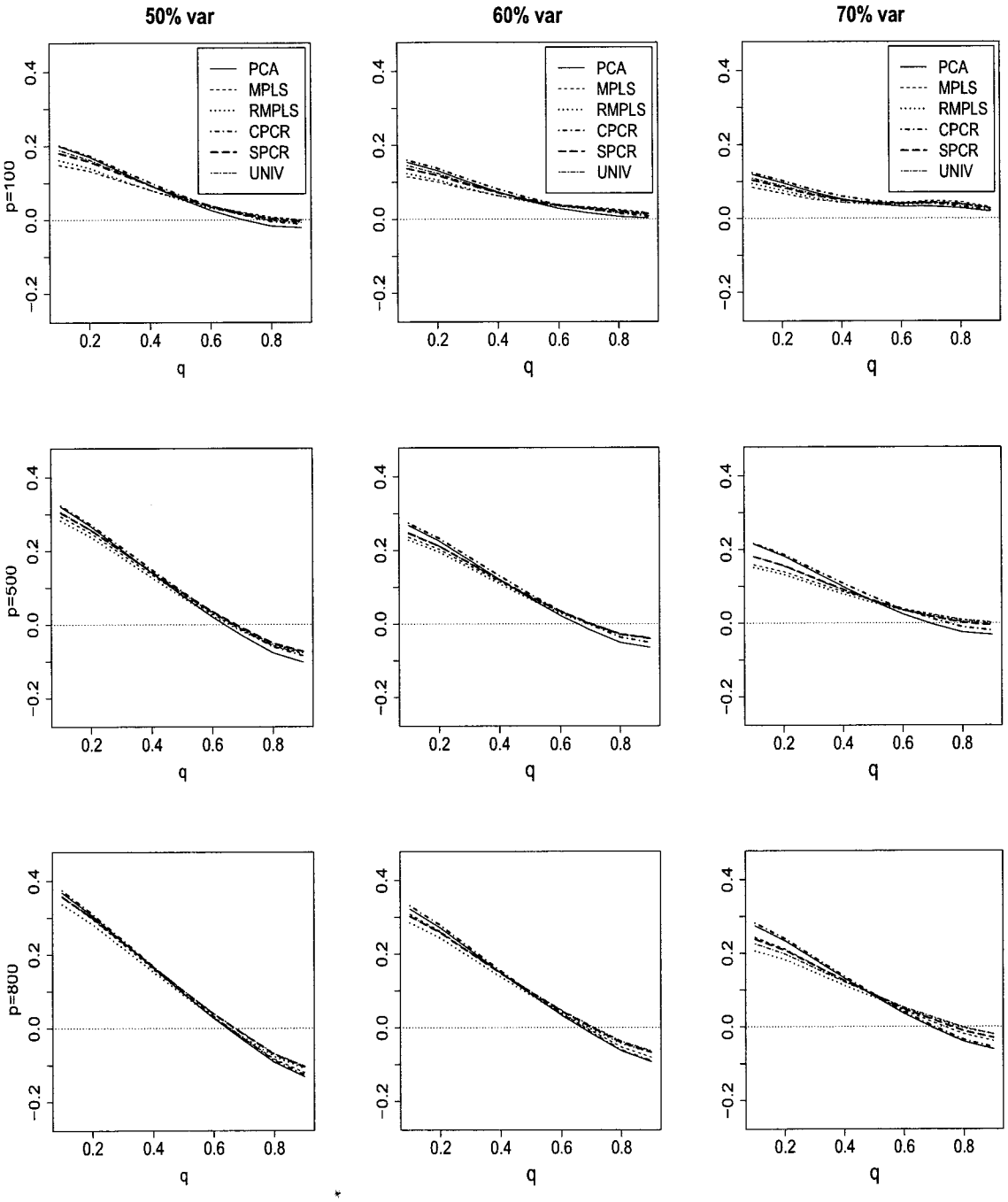


Figure B.4 : Cox model: 1/3 censored. The mean squared error of the estimated weights on the genes $MSE(\beta)$, mean squared error of the estimated survival function evaluated at the average of the covariates $ave(d^2)$, and the mean squared error of the estimated survival function evaluated at the covariates of the individuals $ave(d^2.ind)$ are plotted for datasets with 50% and 70% TVPE accounted for by the first 3 PCs comparing PCA, MPLS, PCA-SIR, and MPLS-SIR. The $x-$axis denotes the number of genes, p. The top row is the plot of the $MSE(\beta)$, middle row is $ave(d^2)$, and the bottom row is $ave(d^2.ind)$.

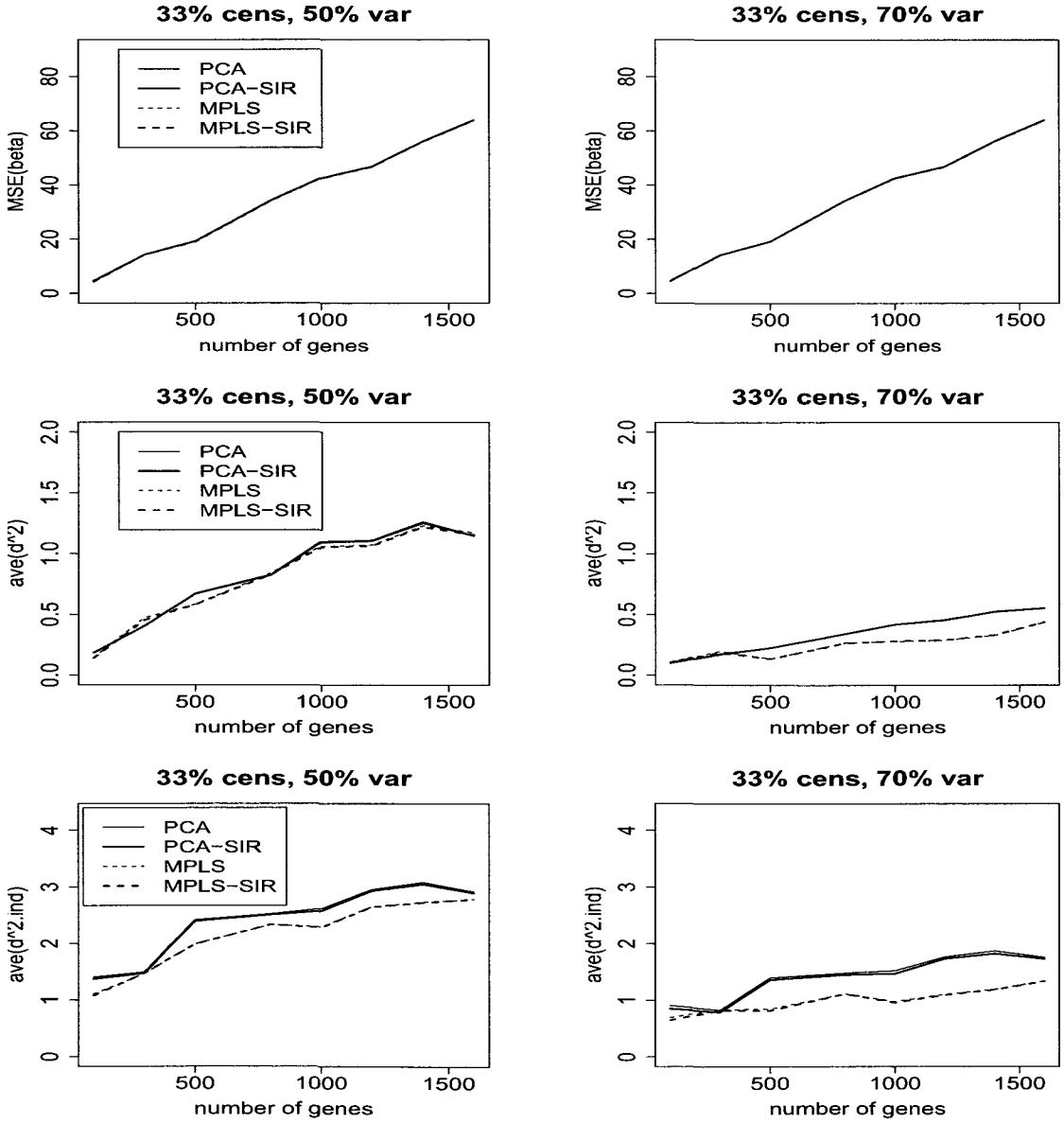


Figure B.5 : Cox model: $r_{ki} \sim Exp(10)$, 1/3 censoring with $p = 100$ and $p = 1000$ for one simulation run. Outliers in the response are present for both $p = 100$ and $p = 1000$. The observed survival times $T_i = min(y_i, c_i)$ are plotted against $X_i'\beta$, where $i = 1, \ldots, N$.

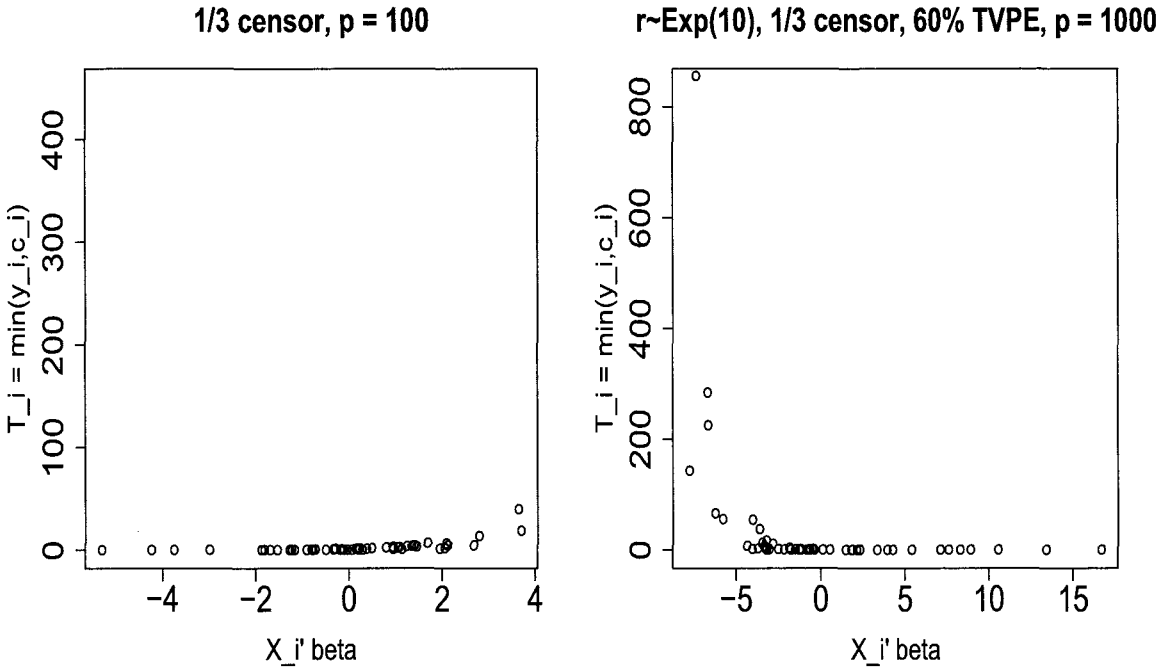
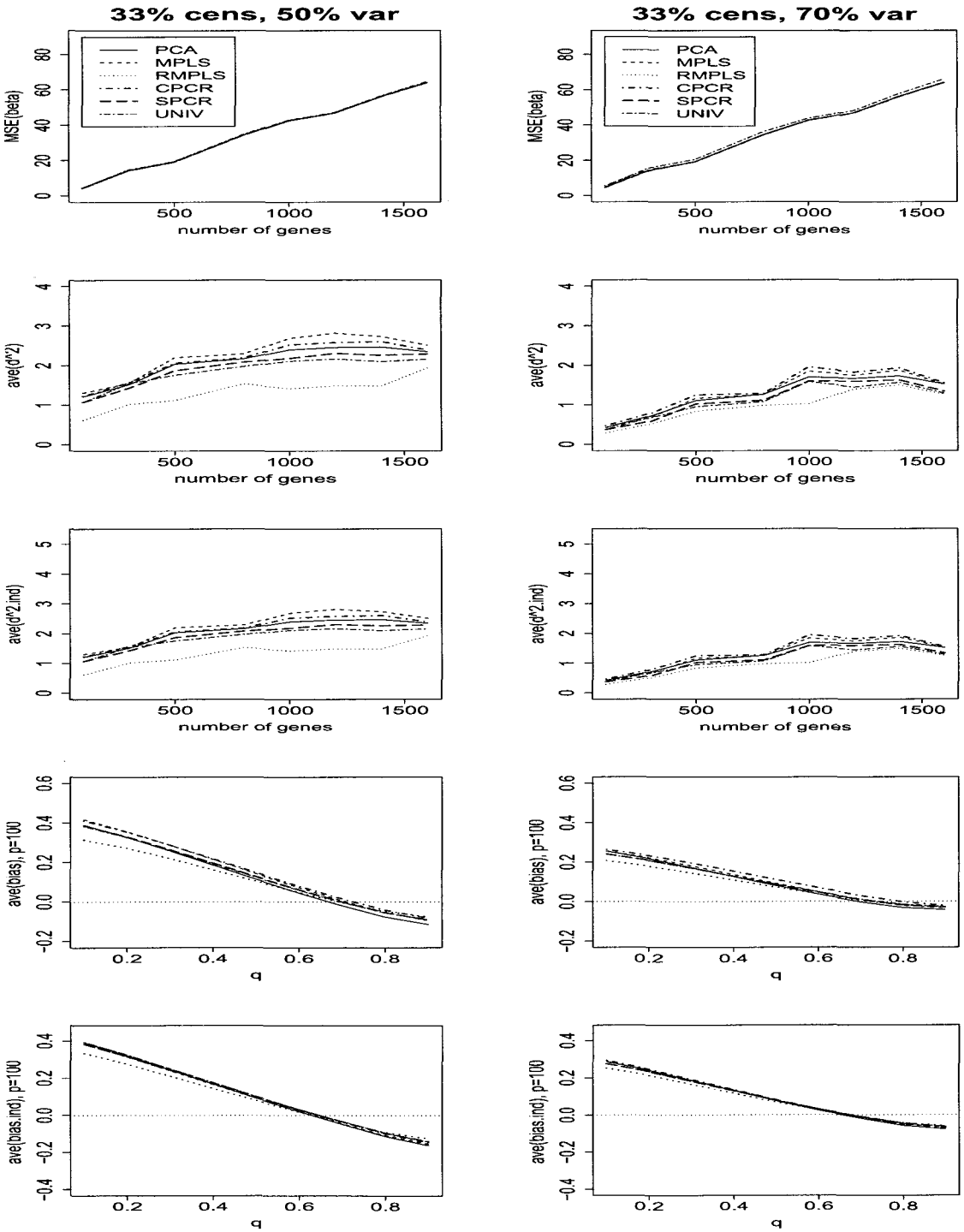


Figure B.6 : Cox model: $r_{ki} \sim Exp(10)$, 1/3 censored. $ave(bias)$ for $p = 100$, and $ave(bias.ind)$ for $p = 100$, for datasets with 50%, and 70% TVPE accounted for by the first 3 PCs comparing PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV based on 5000 simulations. Left panel: 50% TVPE, right panel: 70% TVPE.



B.2 Simulation: Accelerated Failure Time (AFT) Model

Figure B.7 : AFT lognormal mixture model: 1/3 censored. k is chosen by cross-validation (CV). The minimized CV of the fit error $min(CV(fit.error))$, mean squared error of the estimated weights on the genes $MSE(\beta)$, and mean squared error of fit $MSE(fit)$ comparing RWPLS, RRWPLS, MIPLS, RMIPLS, MPLS, and RMPLS (top row), and comparing PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV (bottom row) based on 5000 simulations are plotted.

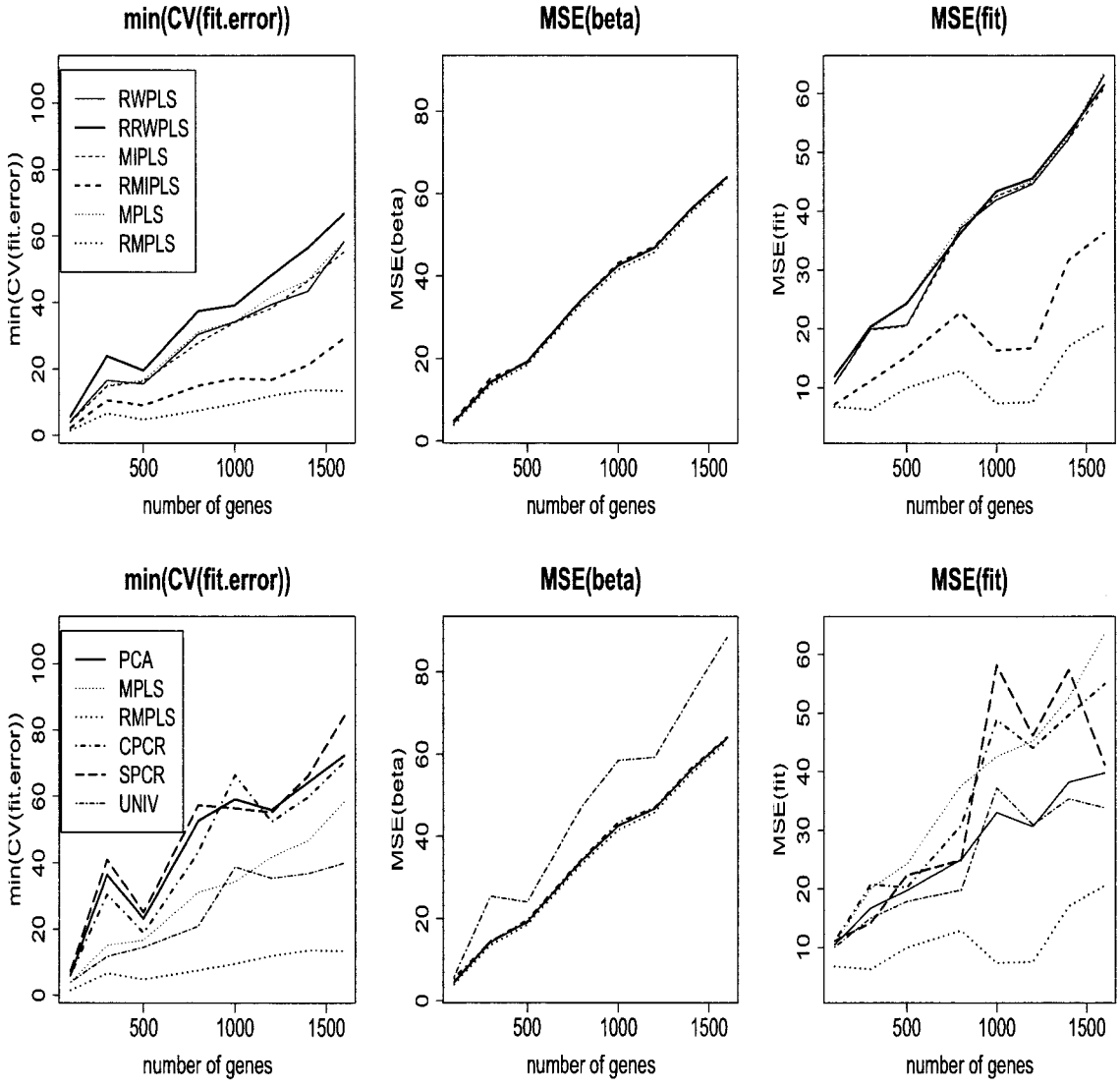


Figure B.8 : AFT lognormal model: 1/3 censored. k is chosen by cross-validation (CV). The minimized CV of the fit error $min(CV(fit.error))$, mean squared error of the estimated weights on the genes $MSE(\beta)$, and mean squared error of fit $MSE(fit)$ comparing RWPLS, RRWPLS, MIPLS, RMIPLS, MPLS, and RMPLS (top row), and comparing PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV (bottom row) based on 5000 simulations are plotted.

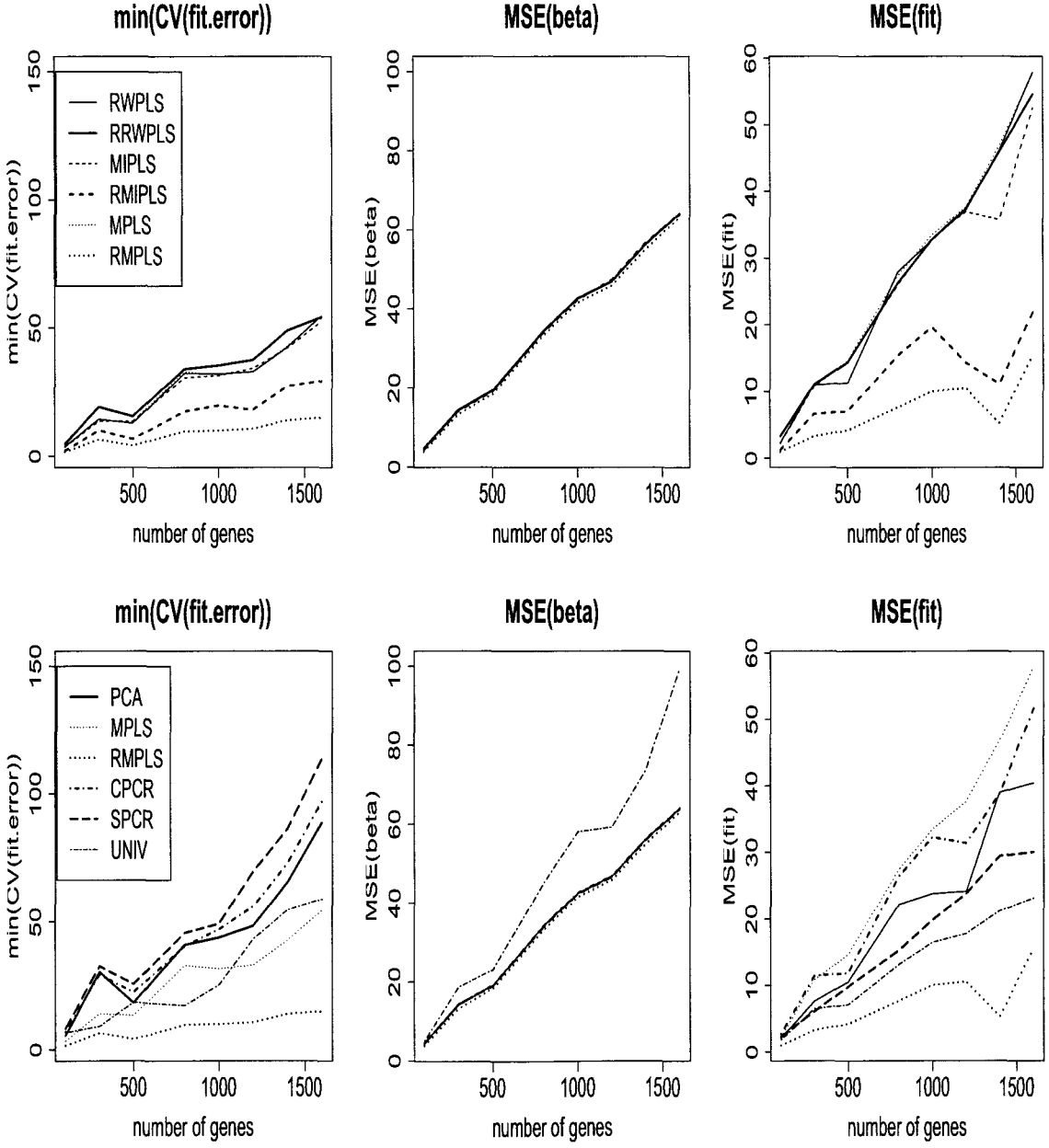
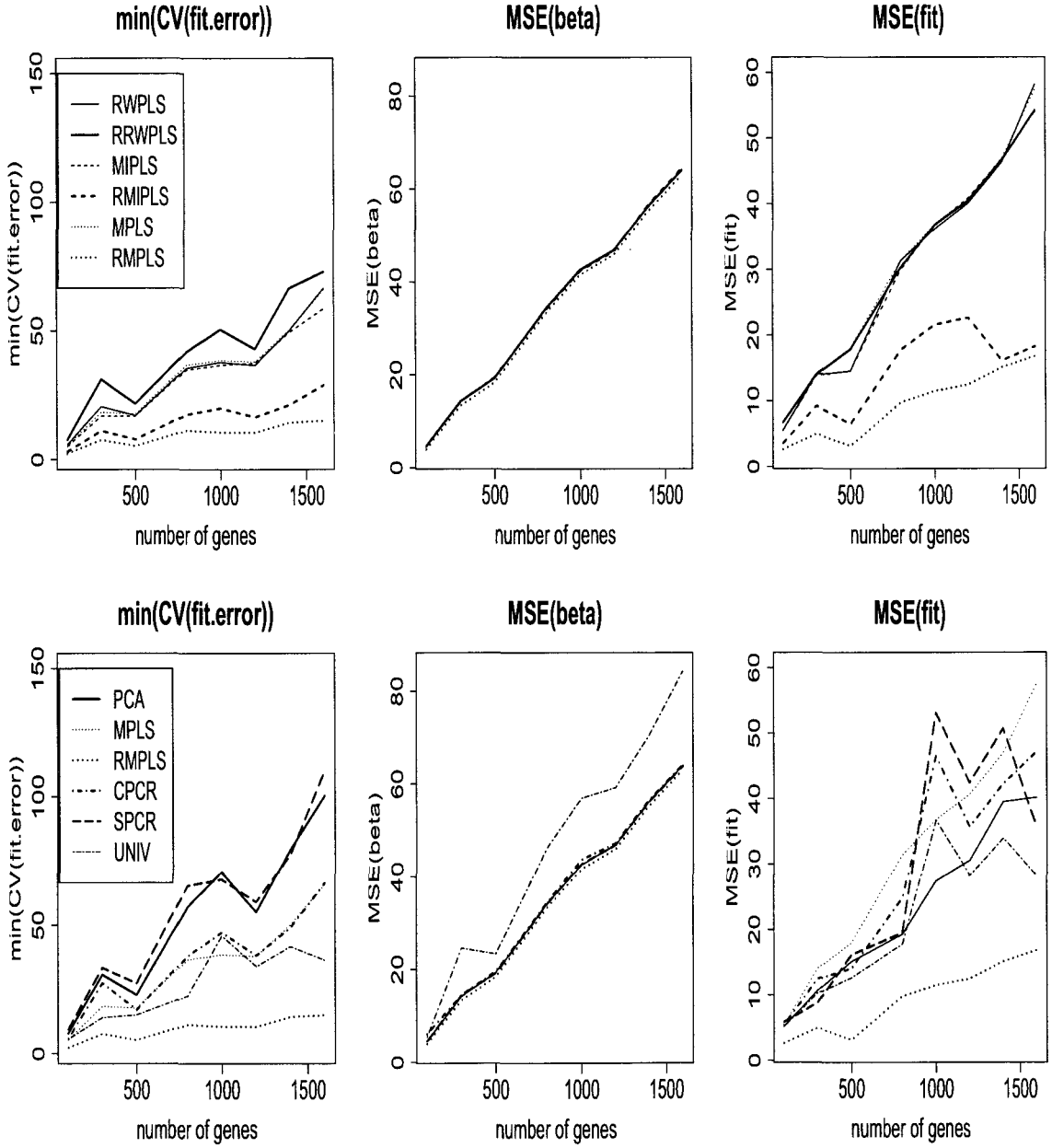


Figure B.9 : AFT logt model: 1/3 censored. k is chosen by cross-validation (CV). The minimized CV of the fit error $min(CV(fit.error))$, mean squared error of the estimated weights on the genes $MSE(\beta)$, and mean squared error of fit $MSE(fit)$ comparing RWPLS, RRWPLS, MIPLS, RMIPLS, MPLS, and RMPLS (top row), and comparing PCA, MPLS, RMPLS, SPCR, CPCR, and UNIV (bottom row) based on 5000 simulations are plotted.



Appendix C

Appendix: Comparison Tables for the Different Methods

C.1 Real Datasets

Table C.1 : Cox model: Number of top-ranked genes in common between MPLS and RMPLS for DLBCL and Harvard datasets using the absolute of the estimated weights for the genes. The first row shows the number of considered top-ranked genes.

K top-ranked genes	25	50	100	250	500	1000
DLBCL	15	33	74	188	397	802
HARVARD	14	28	58	173	369	819

Table C.2 : AFT lognormal model: DLBCL, Harvard, Michigan and Duke datasets. k chosen by CV for the different methods. The minimized cross-validation of the squared fit error $min(CV(fit.error))$ comparing RMPLS to other leading dimension reduction methods, and its standard error of the 1000 repeated runs are shown. RMPLS outperforms other considered methods.

	DLBCL			**HARVARD**			**MICHIGAN**			**DUKE**		
Method	k	error	SE	k	error	SE	k	error	SE	k	error	SE
PCA	5	4.2777	0.613	8	1.6818	0.4334	5	3.0234	0.571	3	7.5555	6.5953
MPLS	3	2.6898	0.3677	1	0.7674	0.1716	2	1.4869	0.5369	1	11.5719	7.6254
RMPLS	3	2.3094	0.3147	1	0.7197	0.1666	3	1.3134	0.4866	2	3.7767	2.7053
RWPLS	1	4.497	0.6638	1	1.4075	0.2928	1	3.7289	0.7274	1	5.7954	1.5946
RRWPLS	1	4.6724	0.6426	1	2.0568	0.4293	1	3.5236	1.1506	1	6.4081	2.5808
MIPLS	3	3.2752	0.3726	1	0.8397	0.1754	1	2.7344	1.2199	2	6.7075	4.358
RMIPLS	3	2.4295	0.3678	1	0.8782	0.2745	1	1.4655	0.4865	1	7.3502	4.3976
CPCR	1	4.9405	1.0879	1	2.0698	0.5926	1	4.5654	2.0616	4	9.8469	8.3316
SPCR	1	4.683	0.934	2	2.9574	1.344	2	4.7596	2.1524	3	15.83	6.4653
UNIV	11	4.8362	1.0441	9	2.4435	1.1892	6	4.6573	2.0643	4	15.1069	8.9731

Table C.3 : AFT log-t model: DLBCL, Harvard, Michigan and Duke datasets. k chosen by CV for the different methods. The $min(CV(fit.error))$ comparing RMPLS to other leading dimension reduction methods, and the standard error of the 1000 repeated runs are shown. RMPLS outperforms other considered methods.

	DLBCL			**HARVARD**			**MICHIGAN**			**DUKE**		
Method	k	error	SE	k	error	SE	k	error	SE	k	error	SE
PCA	4	5.4655	0.772	7	1.7788	0.4683	6	4.8945	0.6511	5	24.3901	7.0841
MPLS	3	2.7872	0.53	3	0.5571	0.2004	3	1.1604	0.4677	1	11.4181	5.6271
RMPLS	6	1.7432	0.3832	4	0.4304	0.1341	3	0.6431	0.2322	2	5.382	3.1963
RWPLS	1	5.8498	0.8737	1	1.8183	0.5998	1	5.4362	0.8935	1	9.819	4.0548
RRWPLS	1	5.7512	0.7406	1	1.9268	0.3576	1	4.6471	1.8023	2	7.1552	2.7309
MIPLS	3	3.0681	0.4645	2	0.6744	0.2211	2	1.8753	0.951	2	9.7551	4.9673
RMIPLS	4	1.948	0.3188	4	0.5926	0.2111	1	1.067	0.3488	1	7.7745	4.783
CPCR	8	4.8893	0.7858	4	1.1819	0.3124	3	3.2976	1.0151	1	9.5548	4.3041
SPCR	1	5.7712	0.8979	1	2.216	0.8825	1	6.4154	2.0608	1	25.7079	9.6766
UNIV	8	4.439	0.6844	4	0.7557	0.2846	7	2.8248	1.0951	5	23.4102	8.7362

Table C.4 : AFT lognormal mixture model: Number of top-ranked genes in common between the ranked versions of PLS and their un-ranked counterparts for DLBCL, Harvard, Michigan and Duke datasets using the absolute of the estimated weights for the genes for 1st component. The first row shows the number of considered top-ranked genes. MPLS and RMPLS shares many genes in common.

	k top-ranked genes	25	50	100	250	500	1000
DLBCL	MPLS and RMPLS	15	33	74	188	397	802
	RWPLS and RRWPLS	0	0	1	32	140	405
	MIPLS and RMIPLS	18	36	76	201	409	843
HARVARD	MPLS and RMPLS	12	28	58	171	368	822
	RWPLS and RRWPLS	0	0	3	15	80	273
	MIPLS and RMIPLS	14	28	69	170	371	804
MICHIGAN	MPLS and RMPLS	10	20	46	117	273	601
	RWPLS and RRWPLS	0	0	0	2	20	126
	MIPLS and RMIPLS	0	0	1	12	45	158
DUKE	MPLS and RMPLS	3	3	3	21	73	210
	RWPLS and RRWPLS	0	3	7	36	105	287
	MIPLS and RMIPLS	0	0	2	18	59	194

Table C.5 : 1/3 censored for $p = 3000$ using Random Projection (RP) with Principal Component Analysis (PCA), and Rank-based Modified Partial Least Squares (RMPLS) under the Cox model. Given $N = 50$, and $\epsilon = .15$, the random projection matrix is of dimension p x $k.r$, where $k.r$ is obtained from the various considered lower bounds for k in this work (the β is ignored for simplicity), then apply PCA or RMPLS to the reduced data matrix. The final reduced data matrix is of dimension N x k, where $k = 5$ is fixed. Denote by JL the RP using the Dasgupta and Gupta bound (Gaussian random matrix), FS the RP using the improved bound obtained from Theorem 3.2 (Gaussian random matrix), ACH the RP using the Achlioptas bound (Rademacher random matrix), and BE the RP using the Berry-Esseen Theorem (Theorem 3.5) (Rademacher random matrix).

		MSE(beta)	**ave(ds.ind)**
PCA		29.2253	2.6802
$JL - PCA$	**k.r = 1546**	29.2366	2.6881
$FS - PCA$	**k.r = 1104**	29.2401	2.6885
$ACH - PCA$	**k.r = 1546**	29.2342	2.6880
$BE - PCA$	**k.r = 1004**	29.2424	2.6868
RMPLS		29.1305	0.9187
$JL - RMPLS$	**k.r = 1546**	29.8197	0.9017
$FS - RMPLS$	**k.r = 1104**	30.0277	0.9153
$ACH - RMPLS$	**k.r = 1546**	29.8056	0.8909
$BE - RMPLS$	**k.r = 1004**	30.0830	0.9260

Biography

Tuan S. Nguyen was born in Saigon, Vietnam on February 28, 1982. He came to the United States at the age of 12. He obtained a bachelor degree in Engineering Mathematics and Statistics from the University of California at Berkeley in 2004, and a master's degree in Statistics from Rice University in 2009. His research interests include Dimension Reduction, Survival Analysis, Microarray Data Analysis, Data Mining, and Clinical Trials. After graduating from Rice University with a Ph.D. degree in Statistics, he will join the late-phase oncology platform at Eli Lilly as a research scientist.